```
FFFFFFFFFFFFFFF     000000000     RRRRRRRRRRRR     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
FFFFFFFFFFFFFFF     000000000     RRRRRRRRRRRR     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
FFFFFFFFFFFFFFF     000000000     RRRRRRRRRRRR     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
FFF               000         000 RRR         RRR RRR         RRR       TTT        LLL
FFF               000         000 RRR         RRR RRR         RRR       TTT        LLL
FFF               000         000 RRR         RRR RRR         RRR       TTT        LLL
FFF               000         000 RRR         RRR RRR         RRR       TTT        LLL
FFF               000         000 RRR         RRR RRR         RRR       TTT        LLL
FFF               000         000 RRR         RRR RRR         RRR       TTT        LLL
FFFFFFFFFFFF      000         000 RRRRRRRRRRRR    RRRRRRRRRRRR          TTT        LLL
FFFFFFFFFFFF      000         000 RRRRRRRRRRRR    RRRRRRRRRRRR          TTT        LLL
FFFFFFFFFFFF      000         000 RRRRRRRRRRRR    RRRRRRRRRRRR          TTT        LLL
FFF               000         000 RRR    RRR      RRR    RRR            TTT        LLL
FFF               000         000 RRR    RRR      RRR    RRR            TTT        LLL
FFF               000         000 RRR    RRR      RRR    RRR            TTT        LLL
FFF               000         000 RRR        RRR  RRR        RRR        TTT        LLL
FFF               000         000 RRR        RRR  RRR        RRR        TTT        LLL
FFF               000         000 RRR        RRR  RRR        RRR        TTT        LLL
FFF                 000000000     RRR            RRR RRR            RRR   TTT        LLLLLLLLLLLLLLL
FFF                 000000000     RRR            RRR RRR            RRR   TTT        LLLLLLLLLLLLLLL
FFF                 000000000     RRR            RRR RRR            RRR   TTT        LLLLLLLLLLLLLLL
```

FILEID**FORUDFWF

```
FFFFFFFFFF    000000    RRRRRRRR    UU      UU  DDDDDDDD    FFFFFFFFFF  WW      WW  FFFFFFFFFF
FFFFFFFFFF    000000    RRRRRRRR    UU      UU  DDDDDDDD    FFFFFFFFFF  WW      WW  FFFFFFFFFF
FF          OO      OO  RR      RR  UU      UU  DD      DD  FF          WW      WW  FF
FF          OO      OO  RR      RR  UU      UU  DD      DD  FF          WW      WW  FF
FF          OO      OO  RR      RR  UU      UU  DD      DD  FF          WW      WW  FF
FF          OO      OO  RR      RR  UU      UU  DD      DD  FF          WW      WW  FF
FFFFFFFF    OO      OO  RRRRRRRR    UU      UU  DD      DD  FFFFFFFF    WW      WW  FFFFFFFF
FFFFFFFF    OO      OO  RRRRRRRR    UU      UU  DD      DD  FFFFFFFF    WW      WW  FFFFFFFF
FF          OO      OO  RR  RR      UU      UU  DD      DD  FF          WW  WW  WW  FF
FF          OO      OO  RR  RR      UU      UU  DD      DD  FF          WW  WW  WW  FF
FF          OO      OO  RR    RR    UU      UU  DD      DD  FF          WWWW  WWWW  FF              ....
FF          OO      OO  RR    RR    UU      UU  DD      DD  FF          WWWW  WWWW  FF              ....
FF            000000    RR      RR  UUUUUUUUUU  DDDDDDDD    FF          WW      WW  FF              ....
FF            000000    RR      RR  UUUUUUUUUU  DDDDDDDD    FF          WW      WW  FF              ....

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

```
;     1          0001  0 MODULE FOR$$UDF_WF (%TITLE 'FORTRAN Write Formatted UDF'
;     2          0002  0                     IDENT = '2-058'          ! File: FORUDFWF.B32  Edit: SBL2058
;     3          0003  0                     ) =
;     4          0004  1 BEGIN
;     5          0005  1 !
;     6          0006  1 !******************************************************************************
;     7          0007  1 !*                                                                            *
;     8          0008  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                   *
;     9          0009  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                     *
;    10          0010  1 !*  ALL RIGHTS RESERVED.                                                       *
;    11          0011  1 !*                                                                            *
;    12          0012  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
;    13          0013  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
;    14          0014  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
;    15          0015  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
;    16          0016  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
;    17          0017  1 !*  TRANSFERRED.                                                               *
;    18          0018  1 !*                                                                            *
;    19          0019  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
;    20          0020  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT     *
;    21          0021  1 !*  CORPORATION.                                                               *
;    22          0022  1 !*                                                                            *
;    23          0023  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
;    24          0024  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                    *
;    25          0025  1 !*                                                                            *
;    26          0026  1 !*                                                                            *
;    27          0027  1 !******************************************************************************
;    28          0028  1
;    29          0029  1 !++
;    30          0030  1 ! FACILITY:  FORTRAN Support Library - not user callable
;    31          0031  1 !
;    32          0032  1 ! ABSTRACT:
;    33          0033  1 !
;    34          0034  1 !       This module implements FORTRAN Write Formatted I/O
;    35          0035  1 !       statements (sequential access - S, direct access - D,
;    36          0036  1 !       ENCODE - M) at the User data Formatter level of
;    37          0037  1 !       abstraction (UDF level is 2nd level). This module
;    38          0038  1 !       calls the Read/Write independent format
;    39          0039  1 !       interpreter (FOR$$FMT_INTRPx) to decode the compiled format
;    40          0040  1 !       statement. This module calls the appropriate write record
;    41          0041  1 !       routine at the record handling level of abstraction (REC
;    42          0042  1 !       level is 3rd level) to write a record.
;    43          0043  1 !
;    44          0044  1 ! ENVIRONMENT:  User access mode; reentrant AST level or not.
;    45          0045  1 !
;    46          0046  1 ! AUTHOR:  Thomas N. Hastings;   CREATION DATE: 20-Feb-77
;    47          0047  1 !
;    48          0048  1 ! MODIFIED BY:
;    49          0049  1 !       Thomas N. Hastings, 12-Mar-77: Version 01
;    50          0050  1 !       Richard Grove, 19-Aug-77: Version 2
;    51          0051  1 ! [Previous edit history removed.  SBL 1-Nov-1982]
;    52          0052  1 ! 2-049 - Instead of checking for a zero ELEM_SIZE to determine an
;    53          0053  1 !         end-of-list call from FOR$$UDF_WF9 use a zero ELEM_TYPE.
;    54          0054  1 !         This allows a zero-length string to be formatted properly.
;    55          0055  1 !         SPR 11-30127  SBL 22-May-1980
;    56          0056  1 ! 2-050 - Convert FOR$$FMT_INTRP1 to JSB linkage.  29-Jul-1981  JAW
;    57          0057  1 ! 2-051 - Use non-character moves when possible to fill buffer to high-
```

```
;    58          0058  1 !           water mark, avoiding a call to BLANK_FILL.  JAW 29-Jul-1981
;    59          0059  1 ! 2-052 - Correct error in edit 2-051.  JAW 05-Aug-1981
;    60          0060  1 ! 2-053 - Combine handling of Hollerith and alphanumeric, to get the
;    61          0061  1 !           benefit of non-character moves for Hollerith, and recast CASE
;    62          0062  1 !           for slightly better code.  JAW 05-Aug-1981
;    63          0063  1 ! 2-054 - Add require file FORMSG.B32 in preparation for enhanced error
;    64          0064  1 !           reporting.  JAW 10-Aug-1981
;    65          0065  1 ! 2-055 - Check for zero-length buffer before changing the carriage
;    66          0066  1 !           control character in DO_WRITE.  JAW 10-Aug-1981
;    67          0067  1 ! 2-056 - Set ISB$V_ERR_OFLO for format codes XE and XG.  SPR 11-38351.
;    68          0068  1 !           JAW 13-Aug-1981
;    69          0069  1 ! 2-057 - Ignore $ if carriage control is not FTN.  JAW 28-Aug-1981
;    70          0070  1 ! 2-058 - Reflect changes needed for separate FORRTL shareable image.  Primarily,
;    71          0071  1 !           we can't have self-relative tables for the conversion routines.
;    72          0072  1 !           SBL 1-Nov-1982
;    73          0073  1 !--
;    74          0074  1
;    75          0075  1 !<BLF/PAGE>
```

```
;    77          0076  1 !
;    78          0077  1 ! PROLOGUE FILE:
;    79          0078  1 !
;    80          0079  1
;    81          0080  1 REQUIRE 'RTLIN:FORPROLOG';                  ! FOR$ definitions
;    82          0146  1 SWITCHES ZIP;                               ! Optimize for speed
;    83          0147  1
;    84          0148  1 !
;    85          0149  1 ! TABLE OF CONTENTS:
;    86          0150  1 !
;    87          0151  1
;    88          0152  1 FORWARD ROUTINE
;    89          0153  1     FOR$$UDF_WF0 : JSB_UDF0 NOVALUE,        ! initialization
;    90          0154  1     FOR$$UDF_WF1 : CALL_CCB NOVALUE,        ! format one user I/O list element
;    91          0155  1     FOR$$UDF_WF9 : JSB_UDF9 NOVALUE,        ! end of user I/O list - finish
;    92          0156  1     BLANK_FILL,                             ! fill string with blanks
;    93          0157  1     MOVE_CHAR : NOVALUE,                    ! move characters
;    94          0158  1     DO_WRITE : JSB_DO_WRITE NOVALUE;        ! do per-record formatting and write
;    95          0159  1
;    96          0160  1 !
;    97          0161  1 ! MACROS:
;    98          0162  1 !
;    99          0163  1
;   100          0164  1 MACRO                                       ! Field definitions for action table
;   101        M 0165  1     WF_EOLST =
;   102          0166  1  0,7,1,0%,                                  ! Check for end of user i/o list
;   103        M 0167  1     WF_CHECKW =
;   104          0168  1  0,6,1,0%,                                  ! Check there are w postions available in output buffer
;   105        M 0169  1     WF_SETDSC =
;   106          0170  1  0,5,1,0%,                                  ! Set up a string descriptor for output field
;   107        M 0171  1     WF_DISPAT =
;   108          0172  1  0,0,4,0%;                                  ! CASE index for dispatch
;   109          0173  1
;   110          0174  1 MACRO                                       ! Attributes-packing macro for attributes table
;   111        M 0175  1     A (E, W, D, NDX) =
;   112          0176  1  (E^7 + W^6 +D^5 + NDX)%;
;   113          0177  1
;   114          0178  1 !
;   115          0179  1 ! EQUATED SYMBOLS:
;   116          0180  1 !
;   117          0181  1 !    NONE
;   118          0182  1 !
;   119          0183  1 ! OWN STORAGE:
;   120          0184  1 !
;   121          0185  1
;   122          0186  1 BIND
;   123          0187  1     WF_ACT =                                ! Action table for UDF_WF1, UDF_WF9 format codes
;   124          0188  1         UPLIT BYTE(
;   125          0189  1
;   126          0190  1 !                    E  C  S
;   127          0191  1 !                    O  H  E
;   128          0192  1 !                    L  E  T
;   129          0193  1 !                    S  C  D
;   130          0194  1 !                    T  K  S
;   131          0195  1 !                       W  C
;   132          0196  1                   A(1,0,0, 0),    ! ER    = 0,    ! 00    ! format syntax error
;   133          0197  1                   A(0,0,0, 0),    ! LP    = 1,    ! 01    ! ( - format reversion point
```

```
;  134     0198  1                A(0,0,0, 0),    ! NLP   = 2,    ! 02    ! n( - left paran of repeat group
;  135     0199  1                A(0,0,0, 0),    ! )     = 3,    ! 03    ! ) - right paren of repeat group
;  136     0200  1                                ! MAINTENANCE NOTE: the above should not be seen by this module, except look
;  137     0201  1                A(1,0,0, 1),    ! EOF   = 4,    ! 04    ! ) - End of format
;  138     0202  1                A(0,0,0, 1),    ! SLS   = 5,    ! 05    ! / - Record separator
;  139     0203  1                A(0,0,0, 2),    ! DLR   = 6,    ! 06    ! $ - Dollar sign: terminal I/O
;  140     0204  1                A(1,0,0, 0),    ! CLN   = 7,    ! 07    ! : - Colon: terminate if end of list
;  141     0205  1                0,0,0,0,        ! UNUSED  8:11
;  142     0206  1                A(0,0,0, 0),    ! _P    = 12,   ! 0C    ! sP - signed scale factor
;  143     0207  1                A(0,0,0, 0),    ! _T    = 13,   ! 0D    ! Tn - Tab Set
;  144     0208  1                                ! The above is seen by lookahead only
;  145     0209  1                A(0,1,0, 4),    ! _X    = 14,   ! 0E    ! nX - Skip n columns
;  146     0210  1                A(0,1,0, 7),    ! _H    = 15,   ! 0F    ! nHcccc - Hollerith
;  147     0211  1                                ! MAINTENANCE NOTE: This routine assumes that
;  148     0212  1                                ! only format codes _A and _H use action 7.
;  149     0213  1                                ! See the CASE ... FROM 0 TO 9.
;  150     0214  1                0,0,            ! UNUSED  16:17
;  151     0215  1                A(0,0,0, 0),    ! TL    = 18    ! 12    ! TLn - Tab left n
;  152     0216  1                A(0,0,0, 0),    ! TR    = 19    ! 13    ! TRn - Tab right n
;  153     0217  1                                ! The above two are seen by lookahead only
;  154     0218  1                A(1,0,0, 6),    ! _Q    = 20,   ! 14    ! Q
;  155     0219  1                A(1,1,0, 7),    ! _A    = 21,   ! 15    ! nAw - Alpha numeric
;  156     0220  1                A(1,1,1, 8),    ! _L    = 22,   ! 16    ! nLw - Logical
;  157     0221  1                A(1,1,1, 8),    ! _O    = 23,   ! 17    ! nOw - Octal
;  158     0222  1                A(1,1,1, 8),    ! _I    = 24,   ! 18    ! nIw - Integer
;  159     0223  1                A(1,1,1, 8),    ! _Z    = 25,   ! 19    ! nZw - Hexadecimal
;  160     0224  1                A(1,1,1, 8),    ! XO    = 26    ! 1A    ! nOw.m
;  161     0225  1                A(1,1,1, 8),    ! XI    = 27    ! 1B    ! nIw.m
;  162     0226  1                A(1,1,1, 8),    ! XZ    = 28    ! 1C    ! nZw.m
;  163     0227  1                0,              ! UNUSED  29
;  164     0228  1                A(1,1,1, 9),    ! _F    = 30,   ! 1E    ! nFw.d - Fixed format
;  165     0229  1                A(1,1,1, 9),    ! _E    = 31,   ! 1F    ! nEw.d - Scientific notation format
;  166     0230  1                A(1,1,1, 9),    ! _G    = 32,   ! 20    ! nGw.d - General format
;  167     0231  1                A(1,1,1, 9),    ! _D    = 33,   ! 21    ! nDw.d - Double Precision format
;  168     0232  1                A(1,1,1, 9),    ! XE    = 34    ! 22    ! nEw.dEe
;  169     0233  1                A(1,1,1, 9),    ! XG    = 35    ! 23    ! nGw.dEe
;  170     0234  1                                ! The following codes are used for lookahead only
;  171     0235  1                0,0,0,0,0,      ! UNUSED  36:40
;  172     0236  1                A(1,0,0, 0),    ! _DA   = 41    ! 29    ! nA - default A
;  173     0237  1                A(1,0,0, 0),    ! _DL   = 42    ! 2A    ! nL - default L
;  174     0238  1                A(1,0,0, 0),    ! _DO   = 43    ! 2B    ! nO - default O
;  175     0239  1                A(1,0,0, 0),    ! _DI   = 44    ! 2C    ! nI - default I
;  176     0240  1                A(1,0,0, 0),    ! _DZ   = 45    ! 2D    ! nZ - default Z
;  177     0241  1                0,0,0,0,        ! UNUSED  46:49
;  178     0242  1                A(1,0,0, 0),    ! _DF   = 50    ! 32    ! nF - default F
;  179     0243  1                A(1,0,0, 0),    ! _DE   = 51    ! 33    ! nE - default E
;  180     0244  1                A(1,0,0, 0),    ! _DG   = 52    ! 34    ! nG - default G
;  181     0245  2                A(1,0,0, 0)     ! _DD   = 53    ! 35    ! nD - default D
;  182     0246  1          ) : VECTOR [54, BYTE];
;  183     0247  1
;  184     0248  1 BIND SPACES = UPLIT('    ');
;  185     0249  1
;  186     0250  1 !+
;  187     0251  1 ! Table of conversion routines for integers, indexed by format code (L,O,I,Z).
;  188     0252  1 !-
;  189     0253  1
;  190     0254  1 OWN
```

```
;  191     0255  1      AA_OUT_FIX: VECTOR [4, LONG];
;  192     0256  1
;  193     0257  1  !+
;  194     0258  1  ! Table of conversion routines for reals, indexed by datatype (F,D,G,H) and
;  195     0259  1  ! by format code (F,E,G,D).  Another table is used to map the DSC$K datatype
;  196     0260  1  ! code into the index for this table.
;  197     0261  1  !-
;  198     0262  1
;  199     0263  1  LITERAL
;  200     0264  1      TYP_F = 0,
;  201     0265  1      TYP_D = 1,
;  202     0266  1      TYP_G = 2,
;  203     0267  1      TYP_H = 3,
;  204     0268  1      FMT_F = 0,
;  205     0269  1      FMT_E = 1,
;  206     0270  1      FMT_G = 2,
;  207     0271  1      FMT_D = 3;
;  208     0272  1
;  209     0273  1  STRUCTURE
;  210     0274  1      FLT_ARRAY_ST [T, F; M, N] =
;  211     0275  1          [M*N*%UPVAL]
;  212     0276  1          (FLT_ARRAY_ST+(T*N+F)*%UPVAL);
;  213     0277  1
;  214     0278  1  OWN
;  215     0279  1      AA_OUT_FLT: FLT_ARRAY_ST [4,4];
;  216     0280  1
;  217     0281  1  !+
;  218     0282  1  ! Table that converts DSC$K datatype codes to TYP_ codes for addressing
;  219     0283  1  ! AA_OUT_FLT.
;  220     0284  1  !-
;  221     0285  1
;  222     0286  1  OWN
;  223     0287  1      DTP_TO_TYP: VECTOR [DSC$K_DTYPE_H+1, BYTE] PSECT (_FOR$CODE)
;  224     0288  1          INITIAL (REP 10 OF BYTE(0),
;  225     0289  1          BYTE(TYP_F),
;  226     0290  1          BYTE(TYP_D),
;  227     0291  1          REP 15 OF BYTE(0),
;  228     0292  1          BYTE(TYP_G),
;  229     0293  1          BYTE(TYP_H));
;  230     0294  1
;  231     0295  1  OWN
;  232     0296  1      CVT_INIT: INITIAL(0);                          ! Initialization flag
;  233     0297  1
;  234     0298  1  !
;  235     0299  1  ! EXTERNAL REFERENCES:
;  236     0300  1  !
;  237     0301  1
;  238     0302  1  EXTERNAL
;  239     0303  1      FOR$$AA_REC_PR0 : VECTOR,                      ! PIC array of record processor
;  240     0304  1                                                     ! procedure-initializations in REC
;  241     0305  1                                                     ! level of abstraction. Indexed by
;  242     0306  1                                                     ! I/O statement type (ISB$B_STTM_TYPE)
;  243     0307  1      FOR$$AA_REC_PR1 : VECTOR,                      ! PIC array of record processor procedures
;  244     0308  1                                                     ! Write a record in REC level of
;  245     0309  1                                                     ! abstraction. Indexed by I/O statement
;  246     0310  1                                                     ! type (ISB$B_STTM_TYPE)
;  247     0311  1      FOR$$AA_REC_PR9 : VECTOR;                      ! PIC array of record processor procedures
```

```
;  248          0312  1
;  249          0313  1                                        ! Write last record in REC level of
;  250          0314  1                                        ! abstraction. Indexed by I/O
;  251          0315  1                                        ! statement type (ISB$B_STTM_TYPE)
;  252          0316  1
;  253          0317  1 EXTERNAL ROUTINE
;  254          0318  1     FOR$$FMT_INTRP0 : JSB_FMT0 NOVALUE,    ! initialize format interpreter
;  255          0319  1     FOR$$FMT_INTRP1 : JSB_FMT1 NOVALUE,    ! get next data format code
;  256          0320  1     ! or input-output format code
;  257          0321  1     ! error # and SIGNAL
;  258          0322  1     FOR$$SIGNAL : NOVALUE,                 ! convert FORTRAN err # to
;  259          0323  1                                            ! VAX error # and SIGNAL
;  260          0324  1     FOR$$SIGNAL_STO : NOVALUE,             ! convert FORTRAN err # to
;  261          0325  1                                            ! VAX error # and SIGNAL_STOP
;  262          0326  1     FOR$CVT_F_TD,                          ! F to text, D format
;  263          0327  1     FOR$CVT_F_TE,                          ! F to text, E format
;  264          0328  1     FOR$CVT_F_TF,                          ! F to text, F format
;  265          0329  1     FOR$CVT_F_TG,                          ! F to text, G format
;  266          0330  1     FOR$CVT_D_TD,                          ! D to text, D format
;  267          0331  1     FOR$CVT_D_TE,                          ! D to text, E format
;  268          0332  1     FOR$CVT_D_TF,                          ! D to text, F format
;  269          0333  1     FOR$CVT_D_TG,                          ! D to text, G format
;  270          0334  1     FOR$CVT_G_TD,                          ! G to text, D format
;  271          0335  1     FOR$CVT_G_TE,                          ! G to text, E format
;  272          0336  1     FOR$CVT_G_TF,                          ! G to text, F format
;  273          0337  1     FOR$CVT_G_TG,                          ! G to text, G format
;  274          0338  1     FOR$CVT_H_TD,                          ! H to text, D format
;  275          0339  1     FOR$CVT_H_TE,                          ! H to text, E format
;  276          0340  1     FOR$CVT_H_TF,                          ! H to text, F format
;  277          0341  1     FOR$CVT_H_TG,                          ! H to text, G format
;  278          0342  1     OTS$CVT_L_TL,                          ! L to text, L format
;  279          0343  1     OTS$CVT_L_TO,                          ! L to text, O format
;  280          0344  1     OTS$CVT_L_TI,                          ! L to text, I format
;  281          0345  1     OTS$CVT_L_TZ;                          ! L to text, Z format
;  282          0346  1
```

```
;  284     0347  1 GLOBAL ROUTINE FOR$$UDF_WF0                    ! Write formatted UDF initialization
;  285     0348  1     : JSB_UDF0 NOVALUE =
;  286     0349  1
;  287     0350  1 !++
;  288     0351  1 ! FUNCTIONAL DESCRIPTION:
;  289     0352  1 !
;  290     0353  1 ! Initialize Write Formatted User data formatter (UDF)
;  291     0354  1 !
;  292     0355  1 ! CALLING SEQUENCE:
;  293     0356  1 !
;  294     0357  1 !       JSB FOR$$UDF_WF0
;  295     0358  1 !
;  296     0359  1 ! FORMAL PARAMETERS:
;  297     0360  1 !
;  298     0361  1 !       NONE
;  299     0362  1 !
;  300     0363  1 ! IMPLICIT INPUTS:
;  301     0364  1 !
;  302     0365  1 !       CCB                     Pointer to current logical unit block
;  303     0366  1 !       ISB$B_STTM_TYPE         I/O statement type code - set by
;  304     0367  1 !                               each I/O statement initialization
;  305     0368  1 !
;  306     0369  1 ! IMPLICIT OUTPUTS:
;  307     0370  1 !
;  308     0371  1 !       LUB$A_BUF_BEG           Adr. of first byte of output data buffer
;  309     0372  1 !       LUB$A_BUF_PTR           Adr. of next byte of output
;  310     0373  1 !                               data buffer
;  311     0374  1 !       LUB$A_BUF_HIGH          Adr. of high water byte in output buffer on this
;  312     0375  1 !                               I/O statement
;  313     0376  1 !       LUB$A_BUF_END           Adr. +1 of last char position allocated
;  314     0377  1 !                               to output buffer
;  315     0378  1 !       AA_OUT_FIX              Integer conversion routine addresses
;  316     0379  1 !       AA_OUT_FLT              Floating conversion routine addresses
;  317     0380  1 !
;  318     0381  1 ! ROUTINE VALUE:
;  319     0382  1 !
;  320     0383  1 !       NONE
;  321     0384  1 !
;  322     0385  1 ! SIDE EFFECTS:
;  323     0386  1 !
;  324     0387  1 !       NONE
;  325     0388  1 !
;  326     0389  1 !--
;  327     0390  1
;  328     0391  2     BEGIN
;  329     0392  2
;  330     0393  2     EXTERNAL REGISTER
;  331     0394  2         CCB : REF $FOR$CCB_DECL;
;  332     0395  2
;  333     0396  2     !+
;  334     0397  2     ! Initialize Record processing level of abstraction.
;  335     0398  2     ! Set pointer to current (LUB$A_BUF_PTR)  and last+1
;  336     0399  2     ! (LUB$A_BUF_END) character position for user data in
;  337     0400  2     ! output buffer
;  338     0401  2     !-
;  339     0402  2
;  340     0403  2     JSB_RECO (FOR$$AA_REC_PRO + .FOR$$AA_REC_PRO [.CCB [ISB$B_STTM_TYPE] - ISB$K_FORSTTYLO + 1]);
```

```
;  341          0404  2
;  342          0405  2      !+
;  343          0406  2      ! Initialize character pointer to first position for user
;  344          0407  2      ! data in output buffer - needed only for T AND $ formats
;  345          0408  2      !-
;  346          0409  2
;  347          0410  2      CCB [LUB$A_BUF_BEG] = .CCB [LUB$A_BUF_PTR];
;  348          0411  2
;  349          0412  2      !+
;  350          0413  2      ! Initialize character pointer to highest position
;  351          0414  2      ! written in user data buffer for this record - needed for
;  352          0415  2      ! T format which can position to the left
;  353          0416  2      !-
;  354          0417  2
;  355          0418  2      CCB [LUB$A_BUF_HIGH] = .CCB [LUB$A_BUF_PTR];
;  356          0419  2
;  357          0420  2      !+
;  358          0421  2      ! Initialize format interpreter
;  359          0422  2      !-
;  360          0423  2
;  361          0424  2      FOR$$FMT_INTRPO ();
;  362          0425  2
;  363          0426  2      !+
;  364          0427  2      ! All other ISB locations and flags have already been
;  365          0428  2      ! initialized to 0 or a specified value by the I/O statement
;  366          0429  2      ! initialization for this I/O statement.
;  367          0430  2      !-
;  368          0431  2
;  369          0432  2      !+
;  370          0433  2      ! Initialize conversion routine tables, if necessary.
;  371          0434  2      !-
;  372          0435  2
;  373          0436  2      IF NOT .CVT_INIT
;  374          0437  2      THEN
;  375          0438  3          BEGIN
;  376          0439  3          AA_OUT_FIX [_L-_L] = OTS$CVT_L_TL;
;  377          0440  3          AA_OUT_FIX [_O-_L] = OTS$CVT_L_TO;
;  378          0441  3          AA_OUT_FIX [_I-_L] = OTS$CVT_L_TI;
;  379          0442  3          AA_OUT_FIX [_Z-_L] = OTS$CVT_L_TZ;
;  380          0443  3          AA_OUT_FLT [TYP_F, FMT_F] = FOR$CVT_F_TF;
;  381          0444  3          AA_OUT_FLT [TYP_F, FMT_E] = FOR$CVT_F_TE;
;  382          0445  3          AA_OUT_FLT [TYP_F, FMT_G] = FOR$CVT_F_TG;
;  383          0446  3          AA_OUT_FLT [TYP_F, FMT_D] = FOR$CVT_F_TD;
;  384          0447  3          AA_OUT_FLT [TYP_D, FMT_F] = FOR$CVT_D_TF;
;  385          0448  3          AA_OUT_FLT [TYP_D, FMT_E] = FOR$CVT_D_TE;
;  386          0449  3          AA_OUT_FLT [TYP_D, FMT_G] = FOR$CVT_D_TG;
;  387          0450  3          AA_OUT_FLT [TYP_D, FMT_D] = FOR$CVT_D_TD;
;  388          0451  3          AA_OUT_FLT [TYP_G, FMT_F] = FOR$CVT_G_TF;
;  389          0452  3          AA_OUT_FLT [TYP_G, FMT_E] = FOR$CVT_G_TE;
;  390          0453  3          AA_OUT_FLT [TYP_G, FMT_G] = FOR$CVT_G_TG;
;  391          0454  3          AA_OUT_FLT [TYP_G, FMT_D] = FOR$CVT_G_TD;
;  392          0455  3          AA_OUT_FLT [TYP_H, FMT_F] = FOR$CVT_H_TF;
;  393          0456  3          AA_OUT_FLT [TYP_H, FMT_E] = FOR$CVT_H_TE;
;  394          0457  3          AA_OUT_FLT [TYP_H, FMT_G] = FOR$CVT_H_TG;
;  395          0458  3          AA_OUT_FLT [TYP_H, FMT_D] = FOR$CVT_H_TD;
;  396          0459  3          CVT_INIT = 1;
;  397          0460  2          END;
```

```
;  398          0461  2     RETURNS;
;  399          0462  1     END;

                                                        ! End of FOR$$UDF_WF0 routine


                                                        .TITLE  FOR$$UDF_WF FORTRAN Write Formatted UDF
                                                        .IDENT  \2-058\

                                                        .PSECT  _FOR$DATA,NOEXE,  PIC,2

                                                00000 AA_OUT_FIX:
                                                        .BLKB   16
                                                00010 AA_OUT_FLT:
                                                        .BLKB   64
                                        00000000  00050 CVT_INIT:
                                                        .LONG   0                                               ;

                                                        .PSECT  _FOR$CODE,NOWRT,  SHR,  PIC,2

44  00  00  00  00  00  00  80  02  01  81  00  00  00  80  00000 P.AAA:   .BYTE   -128, 0, 0, 0, -127, 1, 2, -128, 0, 0, 0, -  ;
00  E8  E8  E8  E8  E8  E8  E8  C7  86  00  00  00  00  47  0000F                  0, 0, 0, 68, 71, 0, 0, 0, 0, -122, -57, -  ;
80  80  80  80  00  00  00  00  00  E9  E9  E9  E9  E9  E9  0001E                  -24, -24, -24, -24, -24, -24, -24, 0, -    ;
                        80  80  80  80  00  00  00  00  80  0002D                  -23, -23, -23, -23, -23, -23, 0, 0, 0, 0, -  ;
                                                                                   0, -128, -128, -128, -128, -128, 0, 0, 0, -  ;
                                                                                   0, -128, -128, -128, -128                    ;
                                                00036           .BLKB   2
                                        20  20  20  20  00038 P.AAB:   .ASCII  \    \                             ;
                                                00# 0003C DTP_TO_TYP:
                                                        .BYTE   0[10]                                           ;
                                                00  00046       .BYTE   0                                       ;
                                                01  00047       .BYTE   1                                       ;
                                                00# 00048       .BYTE   0[15]                                   ;
                                                02  00057       .BYTE   2                                       ;
                                                03  00058       .BYTE   3                                       ;

                                        WF_ACT=                         P.AAA
                                        SPACES=                         P.AAB
                                                        .EXTRN  FOR$$AA_REC_PRO
                                                        .EXTRN  FOR$$AA_REC_PR1
                                                        .EXTRN  FOR$$AA_REC_PR9
                                                        .EXTRN  FOR$$FMT_INTRP0
                                                        .EXTRN  FOR$$FMT_INTRP1
                                                        .EXTRN  FOR$$SIGNAL, FOR$$SIGNAL_STO
                                                        .EXTRN  FOR$CVT_F_TD, FOR$CVT_F_TE
                                                        .EXTRN  FOR$CVT_F_TF, FOR$CVT_F_TG
                                                        .EXTRN  FOR$CVT_D_TD, FOR$CVT_D_TE
                                                        .EXTRN  FOR$CVT_D_TF, FOR$CVT_D_TG
                                                        .EXTRN  FOR$CVT_G_TD, FOR$CVT_G_TE
                                                        .EXTRN  FOR$CVT_G_TF, FOR$CVT_G_TG
                                                        .EXTRN  FOR$CVT_H_TD, FOR$CVT_H_TE
                                                        .EXTRN  FOR$CVT_H_TF, FOR$CVT_H_TG
                                                        .EXTRN  OTS$CVT_L_TL, OTS$CVT_L_TO
                                                        .EXTRN  OTS$CVT_L_TI, OTS$CVT_L_TZ

                50      FF71    CB  9A 00000 FOR$$UDF_WF0::
                                                MOVZBL  -143(CCB), R0                                   ; 0403
                50 0000000000G0040  D0 00005    MOVL    FOR$$AA_REC_PRO[R0], R0                         ;
                   0000000000G0040  16 0000D    JSB     FOR$$AA_REC_PRO[R0]                             ;
```

```
                     BC   AB        B0   AB  D0 00014          MOVL    -80(CCB), -68(CCB)                          ; 0410
                     C0   AB        B0   AB  D0 00019          MOVL    -80(CCB), -64(CCB)                          ; 0418
                              00000000G   00  16 0001E          JSB     FOR$$FMT_INTRPO                             ; 0424
                          01 00000000'    EF  E9 00024          BLBC    CVT_INIT, 1$                                ; 0436
                                              05 0002B          RSB                                                 ;
                00000000'    EF 00000000G   00  9E 0002C 1$:    MOVAB   OTS$CVT_L_TL, AA_OUT_FIX                    ; 0439
                00000000'    EF 00000000G   00  9E 00037        MOVAB   OTS$CVT_L_TO, AA_OUT_FIX+4                  ; 0440
                00000000'    EF 00000000G   00  9E 00042        MOVAB   OTS$CVT_L_TI, AA_OUT_FIX+8                  ; 0441
                00000000'    EF 00000000G   00  9E 0004D        MOVAB   OTS$CVT_L_TZ, AA_OUT_FIX+12                 ; 0442
                00000000'    EF 00000000G   00  9E 00058        MOVAB   FOR$CVT_F_TF, AA_OUT_FLT                    ; 0443
                00000000'    EF 00000000G   00  9E 00063        MOVAB   FOR$CVT_F_TE, AA_OUT_FLT+4                  ; 0444
                00000000'    EF 00000000G   00  9E 0006E        MOVAB   FOR$CVT_F_TG, AA_OUT_FLT+8                  ; 0445
                00000000'    EF 00000000G   00  9E 00079        MOVAB   FOR$CVT_F_TD, AA_OUT_FLT+12                 ; 0446
                00000000'    EF 00000000G   00  9E 00084        MOVAB   FOR$CVT_D_TF, AA_OUT_FLT+16                 ; 0447
                00000000'    EF 00000000G   00  9E 0008F        MOVAB   FOR$CVT_D_TE, AA_OUT_FLT+20                 ; 0448
                00000000'    EF 00000000G   00  9E 0009A        MOVAB   FOR$CVT_D_TG, AA_OUT_FLT+24                 ; 0449
                00000000'    EF 00000000G   00  9E 000A5        MOVAB   FOR$CVT_D_TD, AA_OUT_FLT+28                 ; 0450
                00000000'    EF 00000000G   00  9E 000B0        MOVAB   FOR$CVT_G_TF, AA_OUT_FLT+32                 ; 0451
                00000000'    EF 00000000G   00  9E 000BB        MOVAB   FOR$CVT_G_TE, AA_OUT_FLT+36                 ; 0452
                00000000'    EF 00000000G   00  9E 000C6        MOVAB   FOR$CVT_G_TG, AA_OUT_FLT+40                 ; 0453
                00000000'    EF 00000000G   00  9E 000D1        MOVAB   FOR$CVT_G_TD, AA_OUT_FLT+44                 ; 0454
                00000000'    EF 00000000G   00  9E 000DC        MOVAB   FOR$CVT_H_TF, AA_OUT_FLT+48                 ; 0455
                00000000'    EF 00000000G   00  9E 000E7        MOVAB   FOR$CVT_H_TE, AA_OUT_FLT+52                 ; 0456
                00000000'    EF 00000000G   00  9E 000F2        MOVAB   FOR$CVT_H_TG, AA_OUT_FLT+56                 ; 0457
                00000000'    EF 00000000G   00  9E 000FD        MOVAB   FOR$CVT_H_TD, AA_OUT_FLT+60                 ; 0458
                00000000'    EF             01  D0 00108        MOVL    #1, CVT_INIT                                ; 0459
                                              05 0010F          RSB                                                 ; 0462

; Routine Size:  272 bytes,    Routine Base:  _FOR$CODE + 0059

;  400          0463  1
```

```
;  402          0464  1 GLOBAL ROUTINE FOR$$UDF_WF1 (                    ! Format one user output element
;  403          0465  1          ELEM_TYPE,                            ! Type code of user I/O list element
;  404          0466  1          ELEM_SIZE,                            ! No. of addressable units in element
;  405          0467  1          ELEM_ADR)                             ! Adr. of element
;  406          0468  1      : CALL_CCB NOVALUE =
;  407          0469  1
;  408          0470  1 !++
;  409          0471  1 ! FUNCTIONAL DESCRIPTION:
;  410          0472  1 !
;  411          0473  1 !       FOR$$UDF_WF1 formats a single user I/O list element
;  412          0474  1 !       and places it in the current output buffer, truncating
;  413          0475  1 !       if necessary to fit. It and the format interpreter,
;  414          0476  1 !       (FOR$$FMT_INTRP1) interprets all format codes until the
;  415          0477  1 !       first I/O list element transmitting format code is
;  416          0478  1 !       encountered.  It then continues executing format codes until
;  417          0479  1 !       lookahead shows that the next format code would be a data
;  418          0480  1 !       transmitter or end-of-list type.
;  419          0481  1 !
;  420          0482  1 !       FOR$$UDF_WF1 is also called when the user I/O list
;  421          0483  1 !       had no elements. This is indicated with .ELEM_TYPE=0
;  422          0484  1 !       FOR$$UDF_WF1 and FOR$$FMT_INTRP1 interpret all format
;  423          0485  1 !       codes up to the first data formatting one, :,
;  424          0486  1 !       or end of format.
;  425          0487  1 !
;  426          0488  1 ! CALLING SEQUENCE:
;  427          0489  1 !
;  428          0490  1 !       CALL FOR$$UDF_WF1 (elem_type.rlu.v, elem_size.rlu.v, elem_adr.rx.r)
;  429          0491  1 !
;  430          0492  1 ! FORMAL PARAMETERS:
;  431          0493  1 !
;  432          0494  1 !       ELEM_TYPE.rlu.v         Type code of user I/O list
;  433          0495  1 !                               element. Form: ELEM_TYPE_x
;  434          0496  1 !                               x = B,W,L,WU,LU,F,D,FC or T.
;  435          0497  1 !                               If zero, this is an end-of-list call.
;  436          0498  1 !       ELEM_SIZE.rlu.v         Size of user I/O list element
;  437          0499  1 !                               in addressable machine units
;  438          0500  1 !       ELEM_ADR.rx.r           Adr. of user I/O list element
;  439          0501  1 !                               x = b, w, l, wu, lu, f, d, fc,
;  440          0502  1 !                                 t, g, h, dc or gc.
;  441          0503  1 !
;  442          0504  1 ! IMPLICIT INPUTS:
;  443          0505  1 !
;  444          0506  1 !       CCB                     Pointer to current logical unit block
;  445          0507  1 !       ISB$B_STTM_TYPE         I/O statement type code - set by each
;  446          0508  1 !                               I/O statement initialization
;  447          0509  1 !
;  448          0510  1 ! The following ISB locations are set only by previous calls to
;  449          0511  1 ! FOR$$UDF_WF(0,1), i.e., are effectively OWN.
;  450          0512  1 !
;  451          0513  1 !       LUB$A_BUF_BEG           Pointer to first char. position in
;  452          0514  1 !                               user data part of output buffer
;  453          0515  1 !       LUB$A_BUF_PTR           Pointer to next char. position
;  454          0516  1 !                               in user data part of output buffer
;  455          0517  1 !       LUB$A_BUF_HIGH          Pointer to highest char. position
;  456          0518  1 !                               written so far on any T format code
;  457          0519  1 !       LUB$A_BUF_END           Pointer to last+1 char. position
;  458          0520  1 !                               in user data part of output buffer
```

```
;  459     0521  1 !       ISB$V_DOLLAR            Dollar sign seen in format for this
;  460     0522  1 !                               record, if 1. Change carriage
;  461     0523  1 !                               control SP (space) to $, + to Null (0).
;  462     0524  1 !   The following ISB locations are set by the format interpreter
;  463     0525  1 !   (FOR$$FMT_INTRP1) which this module calls:
;  464     0526  1 !
;  465     0527  1 !       ISB$A_FMT_PTR           Pointer to next char. position
;  466     0528  1 !                               in user data part of output buffer
;  467     0529  1 !                               Used only in H format.
;  468     0530  1 !       ISB$W_FMT_W             Field width (w)
;  469     0531  1 !       ISB$B_FMT_D             No. of fraction digits (d)
;  470     0532  1 !       ISB$B_FMT_E             No. of exponent characters (e)
;  471     0533  1 !       ISB$B_FMT_P             Signed scale factor (p)
;  472     0534  1 !
;  473     0535  1 ! IMPLICIT OUTPUTS:
;  474     0536  1 !
;  475     0537  1 !       ISB$A_FMT_PTR           Pointer to next char. position
;  476     0538  1 !                               in compiled format character string
;  477     0539  1 !                               Changed only for H format.
;  478     0540  1 !
;  479     0541  1 ! The following ISB locations are set only by previous calls
;  480     0542  1 ! to FOR$$UDF_WF{0,1}, i.e., are effectively OWN.
;  481     0543  1 !
;  482     0544  1 !       LUB$A_BUF_PTR           Pointer to next char. position
;  483     0545  1 !                               in user data part of output buffer
;  484     0546  1 !       LUB$A_BUF_HIGH          Pointer to highest char. position
;  485     0547  1 !                               written so far on any T format code
;  486     0548  1 !       ISB$V_DOLLAR            Dollar sign seen in format for this
;  487     0549  1 !                               record, if 1. Change carriage
;  488     0550  1 !                               control SP to $, + to Null.
;  489     0551  1 !
;  490     0552  1 ! FUNCTIONAL VALUE:
;  491     0553  1 !
;  492     0554  1 !       NONE
;  493     0555  1 !
;  494     0556  1 ! SIDE EFFECTS:
;  495     0557  1 !
;  496     0558  1 !       SIGNAL_STOPs FOR$_OUTSTAOVE (66='OUTPUT STATEMENT OVERFLOWED RECORD')
;  497     0559  1 !       if user attampts to write beyond the end of the record buffer.
;  498     0560  1 !       SIGNALS FOR$_OUTCONERR (63='OUTPUT CONVERSION ERROR') -
;  499     0561  1 !           overflowed field is filled with *'s.
;  500     0562  1 !       SIGNALS FOR$_FORVARMIS (61='FORMAT/VARIABLE-TYPE MISMATCH')
;  501     0563  1 !--
;  502     0564  1
;  503     0565  2     BEGIN
;  504     0566  2
;  505     0567  2     EXTERNAL REGISTER
;  506     0568  2         CCB : REF $FOR$CCB_DECL;
;  507     0569  2
;  508     0570  2     MAP
;  509     0571  2         ELEM_ADR : REF VECTOR;                  ! element is call-by-reference
;  510     0572  2
;  511     0573  2     GLOBAL REGISTER
;  512     0574  2         EL_SIZE = 10,                           ! Element size
;  513     0575  2         DT_SEEN = 9,                            ! Data transmitter seen
;  514     0576  2         FMT_CODE = 8 : BLOCK [1, LONG];         ! Format code
;  515     0577  2
```

```
;  516          0578  2      LOCAL
;  517          0579  2          ACT : BLOCK [1, LONG],                  ! Action table entry for format code
;  518          0580  2          BUF_PTR,                                ! Output buffer pointer from ISB
;  519          0581  2          FMT_W,                                  ! Output field width from ISB
;  520          0582  2          DSC : BLOCK [8, BYTE];                  ! Static string desciptor for output field
;  521          0583  2
;  522          0584  2      EL_SIZE = .ELEM_SIZE;
;  523          0585  2
;  524          0586  2      !+
;  525          0587  2      ! If ELEM_TYPE is zero, then we must be in end-of-list processing.
;  526          0588  2      ! If so, set DT_SEEN to 1 so that we won't try executing a data
;  527          0589  2      ! transmitter.  If not, set DT_SEEN to zero.
;  528          0590  2      !-
;  529          0591  2
;  530          0592  2      IF .ELEM_TYPE EQL 0 THEN DT_SEEN = 1 ELSE DT_SEEN = 0;
;  531          0593  2
;  532          0594  2      !+
;  533          0595  2      ! Perform loop beginning with a call to the format
;  534          0596  2      ! interpreter and continue processing until we get
;  535          0597  2      ! a format code for transmitting the user I/O list data
;  536          0598  2      ! element (i.e., Q,A,L,O,Z,I,F,E,G,D) in which case perform
;  537          0599  2      ! the output conversion and return to the user program.
;  538          0600  2      ! For other formats which do output without reference to
;  539          0601  2      ! the user I/O list, perform output formatting and continue
;  540          0602  2      ! loop (i.e., EOF, /, $, :, T, X, H)
;  541          0603  2      !-
;  542          0604  2
;  543          0605  2      WHILE 1 DO
;  544          0606  2
;  545          0607  2      !+
;  546          0608  2      ! Get next format code requiring output interpretation:
;  547          0609  2      !    1. If repeating an explicit format code, the code
;  548          0610  2      !       is simply obtained from the B_FMT_CODE field of the ISB.
;  549          0611  2      !
;  550          0612  2      !    2. In other cases it is necessary to call FOR$$FMT_INTRP1
;  551          0613  2      !
;  552          0614  2      ! Dispatch on format code and select appropriate actions.
;  553          0615  2      !-
;  554          0616  2
;  555          0617  3          BEGIN
;  556          0618  3
;  557          0619  3          IF .CCB [ISB$W_FMT_REP] GTR 1 AND .CCB [ISB$B_FMT_CODE] LSSU _DA
;  558          0620  3          THEN
;  559          0621  4              BEGIN
;  560          0622  4              FMT_CODE = .CCB [ISB$B_FMT_CODE];
;  561          0623  4              ACT = .WF_ACT [.FMT_CODE];
;  562          0624  4
;  563          0625  4              IF .DT_SEEN
;  564          0626  4              THEN
;  565          0627  4
;  566          0628  4                  IF .ACT [WF_EOLST] THEN EXITLOOP;
;  567          0629  4
;  568          0630  4              CCB [ISB$W_FMT_REP] = .CCB [ISB$W_FMT_REP] - 1;
;  569          0631  4              END
;  570          0632  3          ELSE
;  571          0633  4              BEGIN
;  572          0634  4
```

```
;  573     0635  4          !+
;  574     0636  4          ! If DT_SEEN is true, then we only want to krow if the next
;  575     0637  4          ! format code would transmit a data item.  Rather than have
;  576     0638  4          ! the high overhead of calling the format interpreter, we
;  577     0639  4          ! can look ahead into the format for this information.  We
;  578     0640  4          ! can't make a 100% determination, so if the format is not
;  579     0641  4          ! an "EOLST" type, call the format interpreter anyway.
;  580     0642  4          ! This is a speed optimization.  If necessary, the code
;  581     0643  4          ! between the "!**"s can be removed with no functionality loss.
;  582     0644  4          !-
;  583     0645  4
;  584     0646  4          !**
;  585     0647  4
;  586     0648  4          IF .DT_SEEN
;  587     0649  4          THEN
;  588     0650  5              BEGIN
;  589     0651  5
;  590     0652  5              LOCAL
;  591     0653  5                  P;                              ! Pointer into format
;  592     0654  5
;  593     0655  5              P = .CCB [ISB$A_FMT_PTR];
;  594     0656  5              FMT_CODE = CH$RCHAR_(.P);           ! Get next format code
;  595     0657  5              FMT_CODE [V_FMT_REPRE] = 0;         ! Clear bit for comparison
;  596     0658  5              ACT = .WF_ACT [.FMT_CODE];
;  597     0659  5
;  598     0660  5              IF .ACT [WF_EOLST] THEN EXITLOOP;       ! End of list type
;  599     0661  5
;  600     0662  4              END;
;  601     0663  4
;  602     0664  4              !**
;  603     0665  4
;  604     0666  4              FOR$$FMT_INTRP1 ();             ! Call format interpreter.
;  605     0667  4                                              ! Implicit arguments are EL_SIZE
;  606     0668  4                                              ! and DT_SEEN.  Implicit result
;  607     0669  4                                              ! is FMT_CODE.
;  608     0670  4
;  609     0671  4              ACT = .WF_ACT [.FMT_CODE];
;  610     0672  4
;  611     0673  4              !+
;  612     0674  4              ! If DT_SEEN was set, and the next format character was
;  613     0675  4              ! an "end of list", the format interpreter returned a format
;  614     0676  4              ! code of zero, without evaluating VFE's or advancing the
;  615     0677  4              ! pointer.  Therefore, if we have now seen a data transmitter
;  616     0678  4              ! and this is an "end of list" format code, we can exit.
;  617     0679  4              !-
;  618     0680  4
;  619     0681  4              IF .DT_SEEN AND .ACT [WF_EOLST] THEN EXITLOOP;
;  620     0682  4
;  621     0683  3              END;
;  622     0684
;  623     0685  3          !+
;  624     0686  3          ! Check for field extending beyond end of output buffer.
;  625     0687  3          ! SIGNAL_STOP FOR$_OUTSTAOVE if the buffer is exceeded.
;  626     0688  3          ! Advance buffer pointer in ISB.
;  627     0689  3          !-
;  628     0690
;  629     0691  3          BUF_PTR = .CCB [LUB$A_BUF_PTR];
```

```
;  630          0692  3          FMT_W = .CCB [ISB$W_FMT_W];
;  631          0693  3
;  632          0694  3          IF .ACT [WF_CHECKW]
;  633          0695  3          THEN
;  634          0696  4              BEGIN
;  635          0697  4              CCB [LUB$A_BUF_PTR] = .BUF_PTR + .FMT_W;
;  636          0698  4
;  637          0699  5              IF (.CCB [LUB$A_BUF_PTR] GTR .CCB [LUB$A_BUF_END])
;  638          0700  4              THEN
;  639          0701  5                  BEGIN
;  640          0702  5                  FOR$$SIGNAL_STO (FOR$K_OUTSTAOVE);
;  641          0703  5                  RETURN;
;  642          0704  4                  END;
;  643          0705  4
;  644          0706  4              !+
;  645          0707  4              ! Fill with blanks between high water mark and here, if
;  646          0708  4              ! "here" is higher.
;  647          0709  4              !-
;  648          0710  4
;  649          0711  5              BEGIN
;  650          0712  5              LOCAL T;
;  651          0713  5              T = .CCB [LUB$A_BUF_HIGH];
;  652          0714  6              IF (.BUF_PTR GTRA .T)
;  653          0715  5              THEN
;  654          0716  6                  BEGIN
;  655          0717  6                  CASE (CH$DIFF (.BUF_PTR, .T)) FROM 1 TO 8 OF
;  656          0718  6                      SET
;  657          0719  6                          [8] :
;  658          0720  7                              BEGIN
;  659          0721  7                              T = CH$MOVE (4, SPACES, .T);
;  660          0722  7                              T = CH$MOVE (4, SPACES, .T);
;  661          0723  6                              END;
;  662          0724  6
;  663          0725  6                          [4] :
;  664          0726  7                              BEGIN
;  665          0727  7                              T = CH$MOVE (4, SPACES, .T);
;  666          0728  6                              END;
;  667          0729  6
;  668          0730  6                          [7] :
;  669          0731  7                              BEGIN
;  670          0732  7                              T = CH$MOVE (4, SPACES, .T);
;  671          0733  7                              T = CH$MOVE (2, SPACES, .T);
;  672          0734  7                              T = CH$MOVE (1, SPACES, .T);
;  673          0735  6                              END;
;  674          0736  6
;  675          0737  6                          [3] :
;  676          0738  7                              BEGIN
;  677          0739  7                              T = CH$MOVE (2, SPACES, .T);
;  678          0740  7                              T = CH$MOVE (1, SPACES, .T);
;  679          0741  6                              END;
;  680          0742  6
;  681          0743  6                          [6] :
;  682          0744  7                              BEGIN
;  683          0745  7                              T = CH$MOVE (4, SPACES, .T);
;  684          0746  7                              T = CH$MOVE (2, SPACES, .T);
;  685          0747  6                              END;
;  686          0748  6
```

```
;  687     0749  6                        [2] :
;  688     0750  7                            BEGIN
;  689     0751  7                            T = CH$MOVE (2, SPACES, .T);
;  690     0752  6                            END;
;  691     0753  6
;  692     0754  6                        [5] :
;  693     0755  7                            BEGIN
;  694     0756  7                            T = CH$MOVE (4, SPACES, .T);
;  695     0757  7                            T = CH$MOVE (1, SPACES, .T);
;  696     0758  6                            END;
;  697     0759  6
;  698     0760  6                        [1] :
;  699     0761  7                            BEGIN
;  700     0762  7                            T = CH$MOVE (1, SPACES, .T);
;  701     0763  6                            END;
;  702     0764  6
;  703     0765  6                        [OUTRANGE] :
;  704     0766  6                            T = BLANK_FILL (CH$DIFF (.BUF_PTR, .T), .T);
;  705     0767  6                    TES;
;  706     0768  6                CCB [LUB$A_BUF_HIGH] = .T + .FMT_W;
;  707     0769  6                END
;  708     0770  6 !
;  709     0771  6 !
;  710     0772  5            ELSE
;  711     0773  5
;  712     0774  5            !+
;  713     0775  5            ! Set new high water mark if any
;  714     0776  5            !-
;  715     0777  5
;  716     0778  5                IF .CCB [LUB$A_BUF_PTR] GTRA .CCB [LUB$A_BUF_HIGH]
;  717     0779  5                THEN
;  718     0780  5                    CCB [LUB$A_BUF_HIGH] = .CCB [LUB$A_BUF_PTR];
;  719     0781  4            END;
;  720     0782  4
;  721     0783  4                !+
;  722     0784  4                ! Construct a string descriptor for output field if necessary.
;  723     0785  4                !-
;  724     0786  4
;  725     0787  4                IF .ACT [WF_SETDSC]
;  726     0788  4                THEN
;  727     0789  5                    BEGIN
;  728     0790  5                    DSC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  729     0791  5                    DSC [DSC$B_CLASS] = DSC$K_CLASS_S;
;  730     0792  5                    DSC [DSC$W_LENGTH] = .FMT_W;
;  731     0793  5                    DSC [DSC$A_POINTER] = .BUF_PTR;
;  732     0794  4                    END;
;  733     0795  4
;  734     0796  3                END;
;  735     0797  3
;  736     0798  3            !+
;  737     0799  3            ! Dispatch to a format-code-specific action
;  738     0800  3            !-
;  739     0801  3
;  740     0802  3            CASE .ACT [WF_DISPAT] FROM 0 TO 9 OF
;  741     0803  3                SET
;  742     0804  3
;  743     0805  3                [0] :
```

```
;  744     0806  3
;  745     0807  3                    !+
;  746     0808  3                    ! No action required on format code (e.g. Colon)
;  747     0809  3                    !-
;  748     0810  3
;  749     0811  3                ;
;  750     0812  3
;  751     0813  3                [1] :
;  752     0814  3
;  753     0815  3                    !+
;  754     0816  3                    ! EOF,SLS
;  755     0817  3                    ! Write record for end of format or / format codes
;  756     0818  3                    ! Do per-record formatting and write record.  Note that
;  757     0819  3                    ! we now allow more than one record on direct writes.
;  758     0820  3                    ! Initialize all output buffer pointer for next record
;  759     0821  3                    ! in this I/O statement, e.g., ISB$A_BUF_(BEG,PTR,END,HIGH)
;  760     0822  3                    ! and ISB$V_DOLLAR = 0.
;  761     0823  3                    !-
;  762     0824  3
;  763     0825  3                    DO_WRITE (FOR$$AA_REC_PR1 + .FOR$$AA_REC_PR1 [.CCB [ISB$B_STTM_TYPE] - ISB$K_FORSTTYLO + 1])
;  764     0826  3
;  765     0827  3                [2] :
;  766     0828  3
;  767     0829  3                    !+
;  768     0830  3                    ! DLR
;  769     0831  3                    ! Dollar sign: set dollar sign encountered in this
;  770     0832  3                    ! record flag (ISB$V_DOLLAR). Checked when record
;  771     0833  3                    ! written to determine whether to change
;  772     0834  3                    ! carriage control for terminal.
;  773     0835  3                    !-
;  774     0836  3
;  775     0837  3                    CCB [ISB$V_DOLLAR] = 1;
;  776     0838  3
;  777     0839  3                [3] :
;  778     0840  3
;  779     0841  3                    !+
;  780     0842  3                    ! No longer used.
;  781     0843  3                    !-
;  782     0844  3
;  783     0845  3                ;
;  784     0846  3
;  785     0847  3                [4] :
;  786     0848  3
;  787     0849  3                    !+
;  788     0850  3                    ! nX: output n spaces (n in FMT_W)
;  789     0851  3                    !-
;  790     0852  3
;  791     0853  3                    BLANK_FILL (.FMT_W, .BUF_PTR);
;  792     0854  3
;  793     0855  3                [5] :
;  794     0856  3
;  795     0857  3                    !+
;  796     0858  3                    ! No longer used.
;  797     0859  3                    !-
;  798     0860  3
;  799     0861  3                    ;
;  800     0862  3
```

```
;  801          0863  3              [6] :
;  802          0864  3
;  803          0865  3                  !+
;  804          0866  3                  ! Q format - ignore on output but use up I/O list element
;  805          0867  3                  ! Just exit loop and return to user program
;  806          0868  3                  !-
;  807          0869  3
;  808          0870  3                  DT_SEEN = 1;
;  809          0871  3
;  810          0872  3              [7] :
;  811          0873  3
;  812          0874  3                  !+
;  813          0875  3                  ! nA (alphanumeric) and nH (Hollerith):
;  814          0876  3                  !
;  815          0877  3                  ! For nA, output right-justified string in field.
;  816          0878  3                  ! Insert leading spaces or truncate on right as
;  817          0879  3                  ! necessary. Then exit loop and return to user program.
;  818          0880  3                  !
;  819          0881  3                  ! For nH, copy n (FMT_W) characters from format to
;  820          0882  3                  ! output buffer.  Update format character pointer.
;  821          0883  3                  !-
;  822          0884  3
;  823          0885  4                  BEGIN
;  824          0886  4
;  825          0887  4                  LOCAL
;  826          0888  4                      ELEM_PTR;
;  827          0889  4
;  828          0890  4                  IF .FMT_CODE EQLU _A            ! Alphanumeric
;  829          0891  4                  THEN
;  830          0892  5                      BEGIN
;  831          0893  5
;  832          0894  5                      ELEM_PTR = .ELEM_ADR;
;  833          0895  5                      IF .EL_SIZE LSSU .FMT_W
;  834          0896  5                      THEN
;  835          0897  5
;  836          0898  5                          !+
;  837          0899  5                          ! User I/O list element is smaller than
;  838          0900  5                          ! field width w (FMT_W). Fill with
;  839          0901  5                          ! leading spaces.
;  840          0902  5                          !-
;  841          0903  5
;  842          0904  6                          BEGIN
;  843          0905  6                          BUF_PTR = BLANK_FILL (.FMT_W - .EL_SIZE, .BUF_PTR);
;  844          0906  6                          FMT_W = .EL_SIZE;
;  845          0907  5                          END;
;  846          0908  5                      DT_SEEN = 1;
;  847          0909  5                      END
;  848          0910  4                  ELSE                            ! Hollerith
;  849          0911  5                      BEGIN
;  850          0912  5                      ELEM_PTR = .CCB [ISB$A_FMT_PTR];
;  851          0913  5                      CCB [ISB$A_FMT_PTR] = .CCB [ISB$A_FMT_PTR] + .FMT_W;
;  852          0914  4                      END;
;  853          0915  4
;  854          0916  4                  !+
;  855          0917  4                  ! Copy the correct number of bytes.  Use non-character
;  856          0918  4                  ! moves if reasonable.
;  857          0919  4                  !-
```

```
;  858     0920  4
;  859     0921  4                 CASE .FMT_W FROM 0 TO 8 OF
;  860     0922  4                     SET
;  861     0923  4
;  862     0924  4                     [8] :
;  863     0925  5                         BEGIN
;  864     0926  5                         COPY_QUAD_A (ELEM_PTR, BUF_PTR);
;  865     0927  4                         END;
;  866     0928  4
;  867     0929  4                     [4] :
;  868     0930  5                         BEGIN
;  869     0931  5                         COPY_LONG_A (ELEM_PTR, BUF_PTR);
;  870     0932  4                         END;
;  871     0933  4
;  872     0934  4                     [7] :
;  873     0935  5                         BEGIN
;  874     0936  5                         COPY_LONG_A (ELEM_PTR, BUF_PTR);
;  875     0937  5                         COPY_WORD_A (ELEM_PTR, BUF_PTR);
;  876     0938  5                         COPY_BYTE_A (ELEM_PTR, BUF_PTR);
;  877     0939  4                         END;
;  878     0940  4
;  879     0941  4                     [3] :
;  880     0942  5                         BEGIN
;  881     0943  5                         COPY_WORD_A (ELEM_PTR, BUF_PTR);
;  882     0944  5                         COPY_BYTE_A (ELEM_PTR, BUF_PTR);
;  883     0945  4                         END;
;  884     0946  4
;  885     0947  4                     [6] :
;  886     0948  5                         BEGIN
;  887     0949  5                         COPY_LONG_A (ELEM_PTR, BUF_PTR);
;  888     0950  5                         COPY_WORD_A (ELEM_PTR, BUF_PTR);
;  889     0951  4                         END;
;  890     0952  4
;  891     0953  4                     [2] :
;  892     0954  5                         BEGIN
;  893     0955  5                         COPY_WORD_A (ELEM_PTR, BUF_PTR);
;  894     0956  4                         END;
;  895     0957  4
;  896     0958  4                     [5] :
;  897     0959  5                         BEGIN
;  898     0960  5                         COPY_LONG_A (ELEM_PTR, BUF_PTR);
;  899     0961  5                         COPY_BYTE_A (ELEM_PTR, BUF_PTR);
;  900     0962  4                         END;
;  901     0963  4
;  902     0964  4                     [1] :
;  903     0965  5                         BEGIN
;  904     0966  5                         COPY_BYTE_A (ELEM_PTR, BUF_PTR);
;  905     0967  4                         END;
;  906     0968  4
;  907     0969  4                     [0] :
;  908     0970  4                         ;
;  909     0971  4
;  910     0972  4                     [OUTRANGE] :
;  911     0973  4                         MOVE_CHAR (.FMT_W, .ELEM_PTR, .BUF_PTR);
;  912     0974  4                     TES;
;  913     0975  4
;  914     0976  3             END;
```

```
;  915     0977  3
;  916     0978  3                [8] :
;  917     0979  3
;  918     0980  3                    !+
;  919     0981  3                    ! All integer formats (L,O,I,Z) output:
;  920     0982  3                    ! 1) Check data type. If user I/O list element is not integer (B,W,L,WU,LU),
;  921     0983  3                    ! and is not O or Z format,
;  922     0984  3                    ! SIGNAL FOR$_FORVARMIS (61='FORMAT VARIABLE-TYPE MISMATCH').
;  923     0985  3                    ! Then exit loop and return to user program.
;  924     0986  3                    !-
;  925     0987  3
;  926     0988  4                    BEGIN
;  927     0989  4
;  928     0990  4                    LOCAL
;  929     0991  4                        S;                          ! No. of addressable units in user I/O list
;  930     0992  4
;  931     0993  4                    ! element.
;  932     0994  4
;  933     0995  4                    !+
;  934     0996  4                    ! Compensate for extended format codes.
;  935     0997  4                    !-
;  936     0998  4
;  937     0999  4                    IF .FMT_CODE GEQU XO
;  938     1000  4                    THEN
;  939     1001  5                        FMT_CODE = .FMT_CODE - (_L + (XO - _O))
;  940     1002  4                    ELSE
;  941     1003  5                        BEGIN
;  942     1004  5                        FMT_CODE = .FMT_CODE - _L;
;  943     1005  5                        CCB [ISB$B_FMT_D] = 1;      ! Digits in integer part
;  944     1006  4                        END;
;  945     1007  4
;  946     1008  5                    IF .ELEM_TYPE GEQU DSC$K_DTYPE_Q AND (.FMT_CODE EQLU (_I - _L) OR .FMT_CODE EQLU (_L - _L))
;  947     1009  4                    THEN
;  948     1010  5                        BEGIN
;  949     1011  5                        CCB [ISB$B_ERR_NO] = FOR$K_FORVARMIS;
;  950     1012  5                        S = %UPVAL;                 ! treat as if long
;  951     1013  5                        END
;  952     1014  4                    ELSE
;  953     1015  4                        S = .EL_SIZE;
;  954     1016  4
;  955     1017  4                    !+
;  956     1018  4                    ! Call appropriate conversion routine.  If it doesn't fit,
;  957     1019  4                    ! signal FOR$_OUTCONERR.
;  958     1020  4                    !-
;  959     1021  4
;  960     1022  4                    IF NOT (.AA_OUT_FIX [.FMT_CODE]) (.ELEM_ADR, DSC, .CCB [ISB$B_FMT_D], .S,
;  961     1023  4                            .CCB [ISB$B_OUT_FLAGS])
;  962     1024  4                    THEN
;  963     1025  4                        CCB [ISB$B_ERR_NO] = FOR$K_OUTCONERR;
;  964     1026  4
;  965     1027  4                    DT_SEEN = 1;
;  966     1028  3                    END;
;  967     1029  3
;  968     1030  3                [9] :
;  969     1031  3
;  970     1032  3                    !+
;  971     1033  3                    ! Determine correct conversion routine for datatype.
```

```
;  972          1034  3                    ! If value is not floating, signal FOR$_FORVARMIS.
;  973          1035  3                    ! Set scale factor and number of integer digits
;  974          1036  3                    ! appropriately and convert.
;  975          1037  3                    !-
;  976          1038  3
;  977          1039  4                    BEGIN
;  978          1040  4
;  979          1041  4                    LOCAL
;  980          1042  4                        SCALE,                      ! True scale factor
;  981          1043  4                        INT_DIGITS;                 ! Number of integer digits
;  982          1044  4
;  983          1045  4                    !+
;  984          1046  4                    ! Adjust format code for extended formats and offset
;  985          1047  4                    ! to first floating format code.  Also set flag
;  986          1048  4                    ! indicating that exponent field width overflow is an
;  987          1049  4                    ! error for extended formats.
;  988          1050  4                    !-
;  989          1051  4
;  990          1052  4                    IF .FMT_CODE GEQU XE
;  991          1053  4                    THEN
;  992          1054  5                        BEGIN
;  993          1055  5                        FMT_CODE = .FMT_CODE - (3 + _F);
;  994          1056  5                        CCB [ISB$V_ERR_OFLO] = 1;
;  995          1057  5                        END
;  996          1058  4                    ELSE
;  997          1059  5                        BEGIN
;  998          1060  5                        FMT_CODE = .FMT_CODE - _F;
;  999          1061  5                        CCB [ISB$V_ERR_OFLO] = 0;
; 1000          1062  4                        END;
; 1001          1063  4
; 1002          1064  4
; 1003          1065  4                    !+
; 1004          1066  4                    ! Now do the conversion.  Set locals for scale factor and
; 1005          1067  4                    ! number of integer digits based on the format code.
; 1006          1068  4                    !-
; 1007          1069  4
; 1008          1070  5                    IF .FMT_CODE EQLU (_F - _F)     ! _F was subtracted above
; 1009          1071  4                    THEN
; 1010          1072  5                        BEGIN
; 1011          1073  5                        SCALE = .CCB [ISB$B_FMT_P];
; 1012          1074  5                        INT_DIGITS = 0;
; 1013          1075  5                        END
; 1014          1076  4                    ELSE
; 1015          1077  5                        BEGIN
; 1016          1078  5                        SCALE = 0;
; 1017          1079  5                        INT_DIGITS = .CCB [ISB$B_FMT_P];
; 1018          1080  4                        END;
; 1019          1081  4
; 1020          1082  4                    !+
; 1021          1083  4                    ! Choose proper conversion routine and do the conversion.
; 1022          1084  4                    ! If not a floating type, then use F_floating conversion.
; 1023          1085  4                    !-
; 1024          1086  4
; 1025          1087  5                        BEGIN
; 1026          1088  5
; 1027          1089  5                        LOCAL
; 1028          1090  5                            CVT_TYPE;
```

```
;  1029          1091  5
;  1030        P 1092  5                   IF NOT ONE_OF (.ELEM_TYPE, DSC$K_DTYPE_F, DSC$K_DTYPE_D,
;  1031          1093  6                       DSC$K_DTYPE_G, DSC$K_DTYPE_H)
;  1032          1094  5                   THEN
;  1033          1095  6                       BEGIN
;  1034          1096  6                       CVT_TYPE = TYP_F;
;  1035          1097  6                       CCB [ISB$B_ERR_NO] = FOR$K_FORVARMIS;
;  1036          1098  6                       END
;  1037          1099  5                   ELSE
;  1038          1100  5                       CVT_TYPE = .DTP_TO_TYP [.ELEM_TYPE];
;  1039          1101  5                   IF NOT (.AA_OUT_FLT [.CVT_TYPE, .FMT_CODE]) (.ELEM_ADR,
;  1040          1102  5                       DSC, .CCB [ISB$B_FMT_D], .SCALE, .INT_DIGITS,
;  1041          1103  5                       .CCB [ISB$B_FMT_E], .CCB [ISB$B_OUT_FLAGS])
;  1042          1104  5                   THEN
;  1043          1105  5                       CCB [ISB$B_ERR_NO] = FOR$K_OUTCONERR;
;  1044          1106  5
;  1045          1107  4                   END;
;  1046          1108  4
;  1047          1109  4               !+
;  1048          1110  4               ! Exit loop and return to user program
;  1049          1111  4               !-
;  1050          1112  4
;  1051          1113  4               DT_SEEN = 1;
;  1052          1114  3               END;                                ! End of F,E,G,D output
;  1053          1115  3           TES;                                    ! End of CASE
;  1054          1116  3
;  1055          1117  2       END;                                        ! End of processing loop
;  1056          1118  2
;  1057          1119  2   RETURN;                                        ! Return from FOR$$UDF_WF1 routine
;  1058          1120  1   END;                                           ! End of FOR$$UDF_WF1


                           07FC 00000          .ENTRY  FOR$$UDF_WF1, Save R2,R3,R4,R5,R6,R7,R8,R9,-  ;0464
                                                       R10
                                                                                                     ;
5E                   08    C2 00002          SUBL2   #8, SP                                        ;
5A             08    AC    D0 00005          MOVL    ELEM_SIZE, EL_SIZE                            ;0584
55             04    AC    D0 00009          MOVL    ELEM_TYPE, R5                                 ;0592
                     03    12 0000D          BNEQ    1$                                            ;
                   0272    31 0000F          BRW     53$                                           ;
                     59    D4 00012  1$:     CLRL    DT_SEEN                                       ;
01             8D    AB    B1 00014  2$:     CMPW    -115(CCB), #1                                 ;0619
                     1D    15 00018          BLEQ    4$                                            ;
29             8F    AB    91 0001A          CMPB    -113(CCB), #41                                ;
                     17    1E 0001E          BGEQU   4$                                            ;
58             8F    AB    9A 00020          MOVZBL  -113(CCB), FMT_CODE                           ;0622
57           FE6E  CF48    9A 00024          MOVZBL  WF_ACT[FMT_CODE], ACT                         ;0623
05                 59    E9 00028          BLBC    DT_SEEN, 3$                                   ;0625
                   57    95 0002D          TSTB    ACT                                           ;0628
                   01    18 0002F          BGEQ    3$                                            ;
                         04 00031          RET                                                   ;
               8D    AB    B7 00032  3$:     DECW    -115(CCB)                                     ;0630
                   2D    11 00035          BRB     6$                                            ;0619
16                 59    E9 00037  4$:     BLBC    DT_SEEN, 5$                                   ;0648
50             80    AB    D0 0003A          MOVL    -128(CCB), P                                  ;0655
```

```
                                58          60  9A 0003E          MOVZBL  (P), FMT_CODE                                   ; 0656
                                58       80 8F  8A 00041          BICB2   #128, FMT_CODE                                  ; 0657
                                57  FE4D CF48  9A 00045          MOVZBL  WF_ACT[FMT_CODE], ACT                           ; 0658
                                           57  95 0004B          TSTB    ACT                                             ; 0660
                                           01  18 0004D          BGEQ    5$
                                               04 0004F          RET
                            00000000G      00  16 00050 5$:      JSB     FOR$$FMT_INTRP1                                 ; 0666
                                57  FE3C CF48  9A 00056          MOVZBL  WF_ACT[FMT_CODE], ACT                           ; 0671
                                05          59  E9 0005C          BLBC    DT_SEEN, 6$                                     ; 0681
                                           57  95 0005F          TSTB    ACT
                                           01  18 00061          BGEQ    6$
                                               04 00063          RET
                                54       B0 AB  D0 00064 6$:      MOVL    -80(CCB), BUF_PTR                               ; 0691
                                56       89 AB  3C 00068          MOVZWL  -119(CCB), FMT_W                                ; 0692
                      03        57          06  E0 0006C          BBS     #6, ACT, 7$                                     ; 0694
                                          0096  31 00070          BRW     21$
                B0 AB           54          56  C1 00073 7$:      ADDL3   FMT_W, BUF_PTR, -80(CCB)                        ; 0697
                          B4 AB       B0 AB  D1 00078          CMPL    -80(CCB), -76(CCB)                              ; 0699
                                          0C  15 0007D          BLEQ    8$
                                7E    42 8F  9A 0007F          MOVZBL  #66, -(SP)                                      ; 0702
                    00000000G   00          01  FB 00083          CALLS   #1, FOR$$SIGNAL_STO
                                               04 0008A          RET                                                     ; 0701
                                50       C0 AB  D0 0008B 8$:      MOVL    -64(CCB), T                                     ; 0713
                                50          54  D1 0008F          CMPL    BUF_PTR, T                                      ; 0714
                                           58  1B 00092          BLEQU   19$
                      51        54          50  C3 00094          SUBL3   T, BUF_PTR, R1                                  ; 0717
                      07        01          51  CF 00098          CASEL   R1, #1, #7
          0020        002C        0038        0044      0009C 9$:      .WORD   17$-9$,-
          001B        0027        0033        003F      000A4          15$-9$,-
                                                                       13$-9$,-
                                                                       11$-9$,-
                                                                       16$-9$,-
                                                                       14$-9$,-
                                                                       12$-9$,-
                                                                       10$-9$
                                           50  DD 000AC          PUSHL   T                                               ; 0766
                                           51  DD 000AE          PUSHL   R1
                    0000V   CF              02  FB 000B0          CALLS   #2, BLANK_FILL
                                           2E  11 000B5          BRB     18$
                            80      FE14    CF  D0 000B7 10$:      MOVL    SPACES, (T)+                                    ; 0721
                            80      FE0F    CF  D0 000BC 11$:      MOVL    SPACES, (T)+                                    ; 0727
                                           22  11 000C1          BRB     18$                                             ; 0717
                            80      FE08    CF  D0 000C3 12$:      MOVL    SPACES, (T)+                                    ; 0732
                            80      FE03    CF  B0 000C8 13$:      MOVW    SPACES, (T)+                                    ; 0739
                                           11  11 000CD          BRB     17$                                             ; 0740
                            80      FDFC    CF  D0 000CF 14$:      MOVL    SPACES, (T)+                                    ; 0745
                            80      FDF7    CF  B0 000D4 15$:      MOVW    SPACES, (T)+                                    ; 0751
                                           0A  11 000D9          BRB     18$                                             ; 0717
                            80      FDF0    CF  D0 000DB 16$:      MOVL    SPACES, (T)+                                    ; 0756
                            80      FDEB    CF  90 000E0 17$:      MOVB    SPACES, (T)+                                    ; 0762
          C0 AB             50              56  C1 000E5 18$:      ADDL3   FMT_W, T, -64(CCB)                              ; 0768
                                           0C  11 000EA          BRB     20$                                             ; 0714
                      C0 AB       B0 AB  D1 000EC 19$:      CMPL    -80(CCB), -64(CCB)                              ; 0778
                                           05  1B 000F1          BLEQU   20$
                      C0 AB       B0 AB  D0 000F3          MOVL    -80(CCB), -64(CCB)                              ; 0780
                  0D        57          05  E1 000F8 20$:      BBC     #5, ACT, 21$                                    ; 0787
                      02 AE       010E    8F  B0 000FC          MOVW    #270, DSC+2                                     ; 0790
```

```
                        6E          56  B0 00102          MOVW    FMT_W, DSC                          ; 0792
                 04     AE          54  D0 00105          MOVL    BUF_PTR, DSC+4                      ; 0793
     53        57       04          00  EF 00109 21$:     EXTZV   #0, #4, ACT, R3                     ; 0802
               09       00          53  CF 0010E          CASEL   R3, #0, #9                          ;
   FF02      0032     0017        FF02     00112 22$:     .WORD   2$-22$,-                            ;
   0045      0172     FF02        0039     0011A                  23$-22$,-                           ;
                      0105        00B9     00122                  24$-22$,-                           ;
                                                                  2$-22$,-                            ;
                                                                  25$-22$,-                           ;
                                                                  2$-22$,-                            ;
                                                                  53$-22$,-                           ;
                                                                  26$-22$,-                           ;
                                                                  39$-22$,-                           ;
                                                                  45$-22$                             ;
                                  FEEB  31 00126          BRW     2$                                  ;
                        50    FF71  CB  9A 00129 23$:     MOVZBL  -143(CCB), R0                       ; 0825
                        50 00000000G0040  D0 0012E        MOVL    FOR$$AA_REC_PR1[R0], R0             ;
                        50 00000000G0040  9E 00136        MOVAB   FOR$$AA_REC_PR1[R0], R0             ;
                                 0000V  30 0013E          BSBW    DO_WRITE                            ;
                                  FED0  31 00141          BRW     2$                                  ;
                 96     AB          04  88 00144 24$:     BISB2   #4, -106(CCB)                       ; 0837
                                  FEC9  31 00148          BRW     2$                                  ;
                                    54  DD 0014B 25$:     PUSHL   BUF_PTR                             ; 0853
                                    56  DD 0014D          PUSHL   FMT_W                               ;
              0000V     CF          02  FB 0014F          CALLS   #2, BLANK_FILL                      ;
                                  FEBD  31 00154          BRW     2$                                  ;
                        15          58  D1 00157 26$:     CMPL    FMT_CODE, #21                       ; 0890
                                    1F  12 0015A          BNEQ    28$                                 ;
                        52    0C    AC  D0 0015C          MOVL    ELEM_ADR, ELEM_PTR                  ; 0894
                        56          5A  D1 00160          CMPL    EL_SIZE, FMT_W                      ; 0895
                                    11  1E 00163          BGEQU   27$                                 ;
                                    54  DD 00165          PUSHL   BUF_PTR                             ; 0905
               7E       56          5A  C3 00167          SUBL3   EL_SIZE, FMT_W, -(SP)               ;
              0000V     CF          02  FB 0016B          CALLS   #2, BLANK_FILL                      ;
                        54          50  D0 00170          MOVL    R0, BUF_PTR                         ;
                        56          5A  D0 00173          MOVL    EL_SIZE, FMT_W                      ; 0906
                        59          01  D0 00176 27$:     MOVL    #1, DT_SEEN                         ; 0908
                                    08  11 00179          BRB     29$                                 ; 0890
                        52    80    AB  D0 0017B 28$:     MOVL    -128(CCB), ELEM_PTR                 ; 0912
                 80     AB          56  C0 0017F          ADDL2   FMT_W, -128(CCB)                    ; 0913
               08       00          56  CF 00183 29$:     CASEL   FMT_W, #0, #8                       ; 0921
   002D      0035     003E        FE8D     00187 30$:     .WORD   2$-30$,-                            ;
   002A      0032     003B        0024     0018F                  38$-30$,-                           ;
                                  001E     00197                  36$-30$,-                           ;
                                                                  34$-30$,-                           ;
                                                                  32$-30$,-                           ;
                                                                  37$-30$,-                           ;
                                                                  35$-30$,-                           ;
                                                                  33$-30$,-                           ;
                                                                  31$-30$                             ;
                                    14  BB 00199          PUSHR   #^M<R2,R4>                          ; 0973
                                    56  DD 0019B          PUSHL   FMT_W                               ;
              0000V     CF          03  FB 0019D          CALLS   #3, MOVE_CHAR                       ;
                                  FE6F  31 001A2          BRW     2$                                  ;
                        84          82  7D 001A5 31$:     MOVQ    (ELEM_PTR)+, (BUF_PTR)+             ; 0926
                                  FE69  31 001A8          BRW     2$                                  ; 0921
                        84          82  D0 001AB 32$:     MOVL    (ELEM_PTR)+, (BUF_PTR)+             ; 0931
```

```
                        FE63  31 001AE          BRW     2$                                    ; 0921
           84             82  D0 001B1 33$:     MOVL    (ELEM_PTR)+, (BUF_PTR)+               ; 0936
           84             82  B0 001B4 34$:     MOVW    (ELEM_PTR)+, (BUF_PTR)+               ; 0943
                          0C  11 001B7          BRB     38$                                   ; 0944
           84             82  D0 001B9 35$:     MOVL    (ELEM_PTR)+, (BUF_PTR)+               ; 0949
           84             82  B0 001BC 36$:     MOVW    (ELEM_PTR)+, (BUF_PTR)+               ; 0955
                        FE52  31 001BF          BRW     2$                                    ; 0921
           84             82  D0 001C2 37$:     MOVL    (ELEM_PTR)+, (BUF_PTR)+               ; 0960
           84             82  90 001C5 38$:     MOVB    (ELEM_PTR)+, (BUF_PTR)+               ; 0966
                        FE49  31 001C8          BRW     2$                                    ; 0921
           1A             58  D1 001CB 39$:     CMPL    FMT_CODE, #26                         ; 0999
                          05  1F 001CE          BLSSU   40$                                   ;
           58             19  C2 001D0          SUBL2   #25, FMT_CODE                         ; 1001
                          07  11 001D3          BRB     41$                                   ;
           58             16  C2 001D5 40$:     SUBL2   #22, FMT_CODE                         ; 1004
      8B   AB             01  90 001D8          MOVB    #1, -117(CCB)                         ; 1005
           09             55  D1 001DC 41$:     CMPL    R5, #9                                ; 1008
                          13  1F 001DF          BLSSU   43$                                   ;
           02             58  D1 001E1          CMPL    FMT_CODE, #2                          ;
                          04  13 001E4          BEQL    42$                                   ;
                          58  D5 001E6          TSTL    FMT_CODE                              ;
                          0A  12 001E8          BNEQ    43$                                   ;
    FF70   CB             3D  90 001EA 42$:     MOVB    #61, -144(CCB)                        ; 1011
           50             04  D0 001EF          MOVL    #4, S                                 ; 1012
                          03  11 001F2          BRB     44$                                   ; 1008
           50             5A  D0 001F4 43$:     MOVL    EL_SIZE, S                            ; 1015
           51 00000000'EF48   D0 001F7 44$:     MOVL    AA_OUT_FIX[FMT_CODE], R1              ; 1022
           7E        94   AB  9A 001FF          MOVZBL  -108(CCB), -(SP)                      ; 1023
                          50  DD 00203          PUSHL   S                                     ; 1022
           7E        8B   AB  9A 00205          MOVZBL  -117(CCB), -(SP)                      ;
                     0C   AE  9F 00209          PUSHAB  DSC                                   ;
                     0C   AC  DD 0020C          PUSHL   ELEM_ADR                              ;
           61             05  FB 0020F          CALLS   #5, (R1)                              ;
           6A             50  E9 00212          BLBC    R0, 52$                               ;
                          6D  11 00215          BRB     53$                                   ; 1027
           22             58  D1 00217 45$:     CMPL    FMT_CODE, #34                         ; 1052
                          09  1F 0021A          BLSSU   46$                                   ;
           58             21  C2 0021C          SUBL2   #33, FMT_CODE                         ; 1055
      94   AB             02  88 0021F          BISB2   #2, -108(CCB)                         ; 1056
                          07  11 00223          BRB     47$                                   ; 1052
           58             1E  C2 00225 46$:     SUBL2   #30, FMT_CODE                         ; 1060
      94   AB             02  8A 00228          BICB2   #2, -108(CCB)                         ; 1061
                          58  D5 0022C 47$:     TSTL    FMT_CODE                              ; 1070
                          08  12 0022E          BNEQ    48$                                   ;
           52        88   AB  98 00230          CVTBL   -120(CCB), SCALE                      ; 1073
                          51  D4 00234          CLRL    INT_DIGITS                            ; 1074
                          06  11 00236          BRB     49$                                   ; 1070
                          52  D4 00238 48$:     CLRL    SCALE                                 ; 1078
           51        88   AB  98 0023A          CVTBL   -120(CCB), INT_DIGITS                 ; 1079
50 00300018   8F          55  78 0023E 49$:     ASHL    R5, #3145752, R0                      ; 1093
                          09  19 00246          BLSS    50$                                   ;
                          50  D4 00248          CLRL    CVT_TYPE                              ; 1096
    FF70   CB             3D  90 0024A          MOVB    #61, -144(CCB)                        ; 1097
                          06  11 0024F          BRB     51$                                   ; 1092
           50      FC7D CF45  9A 00251 50$:     MOVZBL  DTP_TO_TYP[R5], CVT_TYPE              ; 1100
           50           6840  DE 00257 51$:     MOVAL   (FMT_CODE)[CVT_TYPE], R0              ; 1101
           50 00000000'EF40   D0 0025B          MOVL    AA_OUT_FLT[R0], R0                    ;
```

```
                        7E    94    AB  9A 00263          MOVZBL  -108(CCB), -(SP)                ; 1103
                        7E    8C    AB  9A 00267          MOVZBL  -116(CCB), -(SP)                ;
                              51    DD  0026B             PUSHL   INT_DIGITS                      ; 1102
                              52    DD  0026D             PUSHL   SCALE                           ;
                        7E    8B    AB  9A 0026F          MOVZBL  -117(CCB), -(SP)                ;
                              14    AE  9F 00273          PUSHAB  DSC                             ; 1101
                              0C    AC  DD 00276          PUSHL   ELEM_ADR                        ;
                        60          07  FB 00279          CALLS   #7, (R0)                        ;
                        05          50  E8 0027C          BLBS    R0, 53$                         ;
                FF70    CB          3F  90 0027F 52$:     MOVB    #63, -144(CCB)                  ; 1105
                        59          01  D0 00284 53$:     MOVL    #1, DT_SEEN                     ; 1113
                                  FD8A  31 00287          BRW     2$                              ; 0605
                                        04 0028A          RET                                     ; 1120

; Routine Size:  651 bytes,    Routine Base:  _FOR$CODE + 0169

; 1059          1121  1
```

```
;  1061          1122  1 ROUTINE DO_WRITE (                          ! do per-record formatting and write record
;  1062          1123  1         FOR$$REC_xn)                        ! adr. or record processing routine
;  1063          1124  1     : JSB_DO_WRITE NOVALUE =
;  1064          1125  1 !+
;  1065          1126  1 !FUNCTIONAL DESCRIPTION:
;  1066          1127  1 !
;  1067          1128  1 !       DO_WRITE is a local routine which performs any per-record
;  1068          1129  1 !       formatting (as distinguished from per I/O list element formatting)
;  1069          1130  1 !       and then output the record by calling the appropriate
;  1070          1131  1 !       record processing routine depending on the statement type
;  1071          1132  1 !       (ISB$BSTTM_TYPE) and formal parameter FOR$$REC_xn which
;  1072          1133  1 !       is either (1) FOR$$REC_x1 if this is not the last record
;  1073          1134  1 !       of the I/o statement or (2) FOR$$REC_x9 if the is the last
;  1074          1135  1 !       record of the I/O statement, i.e., this is the end of I/O list call.
;  1075          1136  1 !       Note: DO_WRITE is also called directly from FOR$$UDF_WF9 on end of
;  1076          1137  1 !       I/O list if at end of format too.  Therefore, all end of
;  1077          1138  1 !       list processing should be kept here in DO_WRITE.
;  1078          1139  1 !
;  1079          1140  1 ! CALLING SEQUENCE:
;  1080          1141  1 !
;  1081          1142  1 !       JSB DO_WRITE (R0=for$$rec_xn.s.ar)
;  1082          1143  1 !
;  1083          1144  1 ! FORMAL PARAMETERS:
;  1084          1145  1 !
;  1085          1146  1 !       FOR$$REC_xn.s.ar        Adr. of record processing routine
;  1086          1147  1 !
;  1087          1148  1 ! IMPLICIT INPUTS:
;  1088          1149  1 !
;  1089          1150  1 !       CCB                     Pointer to current logical unit block
;  1090          1151  1 !
;  1091          1152  1 ! The following locations are set only by previous calls to
;  1092          1153  1 ! FOR$$UDF_WF(0,1), i.e., are effectively OWN for this module.
;  1093          1154  1 !
;  1094          1155  1 !       LUB$A_BUF_BEG           Pointer to first char. position in
;  1095          1156  1 !                               user data part of output buffer
;  1096          1157  1 !       LUB$A_BUF_PTR           Pointer to next char. position
;  1097          1158  1 !                               in user data part of output buffer
;  1098          1159  1 !       LUB$A_BUF_HIGH          Pointer to highest char. position
;  1099          1160  1 !                               written so far on any T format code
;  1100          1161  1 !       LUB$A_BUF_END           Pointer to last+1 char. position
;  1101          1162  1 !                               in user data part of output buffer
;  1102          1163  1 !       ISB$V_DOLLAR            Dollar sign seen in format for this
;  1103          1164  1 !                               record, if 1. Change carriage
;  1104          1165  1 !                               control SP to $, + to Null.
;  1105          1166  1 !
;  1106          1167  1 ! IMPLICIT OUTPUTS:
;  1107          1168  1 !
;  1108          1169  1 ! The following locations are set only by previous calls
;  1109          1170  1 ! to FOR$$UDF_WF(0,1), i.e., are effectively OWN for this module.
;  1110          1171  1 !
;  1111          1172  1 !       LUB$A_BUF_BEG           Pointer: set to first char. position
;  1112          1173  1 !                               of next output buffer to be filled.
;  1113          1174  1 !       LUB$A_BUF_PTR           Pointer: set to first char. position
;  1114          1175  1 !                               in user data part of output buffer to be filled
;  1115          1176  1 !       LUB$A_BUF_HIGH          Pointer: set to first char. position
;  1116          1177  1 !                               of user data part of output buffer to be filled
;  1117          1178  1 !       ISB$V_DOLLAR            Set to 0
```

```
;  1118          1179  1 !--
;  1119          1180  1
;  1120          1181  2     BEGIN
;  1121          1182  2
;  1122          1183  2     EXTERNAL REGISTER
;  1123          1184  2         CCB : REF $FOR$CCB_DECL;
;  1124          1185  2
;  1125          1186  2     !+
;  1126          1187  2     ! 1) IF $ seen in format for current record (ISB$V_DOLLAR=1),
;  1127          1188  2     ! and carriage control is FORTRAN, and buffer contains at least one
;  1128          1189  2     ! character, change carriage control character space to $
;  1129          1190  2     ! or + to Null for terminal dialog no CR's
;  1130          1191  2     ! and/or LF's.
;  1131          1192  2     !-
;  1132          1193  2
;  1133          1194  2     IF .CCB [ISB$V_DOLLAR]
;  1134          1195  2     THEN
;  1135          1196  2         IF .CCB [LUB$V_FTN]
;  1136          1197  2         THEN
;  1137          1198  2             IF .CCB [LUB$A_BUF_END] - .CCB [LUB$A_BUF_BEG] GTR 0
;  1138          1199  2             THEN
;  1139          1200  3                 BEGIN
;  1140          1201  3
;  1141          1202  3                 IF CH$RCHAR (.CCB [LUB$A_BUF_BEG]) EQL %C' ' THEN CH_WCHAR (.CCB [LUB$A_BUF_BEG]) = %C'$';
;  1142          1203  3
;  1143          1204  3                 IF CH$RCHAR (.CCB [LUB$A_BUF_BEG]) EQL %C'+' THEN CH_WCHAR (.CCB [LUB$A_BUF_BEG]) = 0;
;  1144          1205  3
;  1145          1206  2                 END;
;  1146          1207  2
;  1147          1208  2     !+
;  1148          1209  2
;  1149          1210  2     !+
;  1150          1211  2     ! 2) Set buffer pointer to the high water mark.  The REC level will
;  1151          1212  2     ! then fill with blanks from there to the end of the buffer.
;  1152          1213  2     !-
;  1153          1214  2
;  1154          1215  2     CCB [LUB$A_BUF_PTR] = .CCB [LUB$A_BUF_HIGH];
;  1155          1216  2     JSB_REC1 (.FOR$$REC_xn);
;  1156          1217  2
;  1157          1218  2     !+
;  1158          1219  2     ! 3) Initialize beginning and highest pointer
;  1159          1220  2     ! (T format) and dollar-sign-seen-this-record flag
;  1160          1221  2     !-
;  1161          1222  2
;  1162          1223  2     CCB [LUB$A_BUF_BEG] = .CCB [LUB$A_BUF_PTR];
;  1163          1224  2     CCB [LUB$A_BUF_HIGH] = .CCB [LUB$A_BUF_PTR];
;  1164          1225  2     CCB [ISB$V_DOL[AR] = 0;
;  1165          1226  2     RETURN;                                     ! Return from DO_WRITE routine
;  1166          1227  1     END;                                        ! End of DO_WRITE routine


                 1E       96   AB            02  E1 00000 DO_WRITE:
                                                          BBC     #2, -106(CCB), 2$                   ; 1194
                                  A0   AB  95 00005       TSTB    -96(CCB)                            ; 1196
```

```
                          19  18 00008         BGEQ    2$                              ;
         51        BC     AB  D0 0000A         MOVL    -68(CCB), R1                    ; 1198
         51        B4     AB  D1 0000E         CMPL    -76(CCB), R1                    ;
                          0F  15 00012         BLEQ    2$                              ;
         20               61  91 00014         CMPB    (R1), #32                       ; 1202
                          03  12 00017         BNEQ    1$                              ;
         61               24  90 00019         MOVB    #36, (R1)                       ;
         2B               61  91 0001C 1$:     CMPB    (R1), #43                       ; 1204
                          02  12 0001F         BNEQ    2$                              ;
                          61  94 00021         CLRB    (R1)                            ;
    B0   AB        C0     AB  D0 00023 2$:     MOVL    -64(CCB), -80(CCB)              ; 1215
                          60  16 00028         JSB     (FOR$$REC_XN)                   ; 1216
    BC   AB        B0     AB  D0 0002A         MOVL    -80(CCB), -68(CCB)              ; 1223
    C0   AB        B0     AB  D0 0002F         MOVL    -80(CCB), -64(CCB)              ; 1224
    96   AB               04  8A 00034         BICB2   #4, -106(CCB)                   ; 1225
                              05 00038         RSB                                     ; 1227

; Routine Size:  57 bytes,    Routine Base:  _FOR$CODE + 03F4

; 1167          1228  1
```

```
;  1169          1229  1 GLOBAL ROUTINE FOR$$UDF_WF9                   ! Formatted output - end of I/O list call
;  1170          1230  1     : JSB_UDF9 NOVALUE =
;  1171          1231  1
;  1172          1232  1 !++
;  1173          1233  1 ! FUNCTIONAL DESCRIPTION:
;  1174          1234  1 !
;  1175          1235  1 !       FOR$$UDF_WF9 performs end of I/O list output formatting.
;  1176          1236  1 !       All format codes are processed until a data transmitting
;  1177          1237  1 !       format code is encountered (or colon) or end of format.
;  1178          1238  1 !
;  1179          1239  1 ! CALLING SEQUENCE:
;  1180          1240  1 !
;  1181          1241  1 !       JSB FOR$$UDF_WF9 ()
;  1182          1242  1 !
;  1183          1243  1 ! FORMAL PARAMETERS:
;  1184          1244  1 !
;  1185          1245  1 !       NONE
;  1186          1246  1 !
;  1187          1247  1 ! IMPLICIT INPUTS:
;  1188          1248  1 !
;  1189          1249  1 !       See FOR$$UDF_WF1
;  1190          1250  1 !
;  1191          1251  1 !
;  1192          1252  1 ! IMPLICIT OUTPUTS:
;  1193          1253  1 !
;  1194          1254  1 !       See FOR$$UDF_WF1
;  1195          1255  1 !
;  1196          1256  1 ! FUNCTION VALUE:
;  1197          1257  1 !
;  1198          1258  1 !       NONE
;  1199          1259  1 !
;  1200          1260  1 ! SIDE EFFECTS:
;  1201          1261  1 !
;  1202          1262  1 !       See FOR$$UDF_WF1
;  1203          1263  1 !--
;  1204          1264  1
;  1205          1265  2     BEGIN
;  1206          1266  2
;  1207          1267  2     EXTERNAL REGISTER
;  1208          1268  2         CCB : REF $FOR$CCB_DECL;
;  1209          1269  2
;  1210          1270  2     !+
;  1211          1271  2     ! If there are no items in I/O list, current format code is 0.
;  1212          1272  2     ! Call data transmit entry point with element type of 0 as
;  1213          1273  2     ! a flag. Return as soon as a data transmitting format code,
;  1214          1274  2     ! colon, or End of Format code is encountered.
;  1215          1275  2     !-
;  1216          1276  2
;  1217          1277  2     IF .CCB [ISB$B_FMT_CODE] EQL 0 THEN FOR$$UDF_WF1 (0, 0, 0);
;  1218          1278  2
;  1219          1279  2     !+
;  1220          1280  2     ! Do the final write
;  1221          1281  2     !-
;  1222          1282  2
;  1223          1283  2     DO_WRITE (FOR$$AA_REC_PR9 + .FOR$$AA_REC_PR9 [.CCB [ISB$B_STTM_TYPE] - ISB$K_FORSTTYLO + 1]);
;  1224          1284  2     RETURN;
;  1225          1285  1     END;                                        ! End of FOR$$UDF_WF9 Routine
```

```
                                 8F    AB  95 00000 FOR$$UDF_WF9::
                                                     TSTB    -113(CCB)                       ; 1277
                                       09  12 00003         BNEQ    1$
                                       7E  7C 00005         CLRQ    -(SP)
                                       7E  D4 00007         CLRL    -(SP)
                              FD2E     CF    03  FB 00009         CALLS   #3, FOR$$UDF_WF1
                                       50  FF71  CB 9A 0000E 1$:      MOVZBL  -143(CCB), R0           ; 1283
                                       50 00000000G0040  D0 00013     MOVL    FOR$$AA_REC_PR9[R0], R0
                                       50 00000000G0040  9E 0001B     MOVAB   FOR$$AA_REC_PR9[R0], R0
                                                 A2  11 00023         BRB     DO_WRITE

; Routine Size:  37 bytes,    Routine Base:  _FOR$CODE + 042D

; 1226          1286  1
```

```
;  1228          1287  1 ROUTINE BLANK_FILL (                          ! Fill a string with blanks
;  1229          1288  1         LEN,                                  ! Fill length
;  1230          1289  1         ADDR)                                 ! Buffer address
;  1231          1290  1     =
;  1232          1291  1
;  1233          1292  1 !++
;  1234          1293  1 ! FUNCTIONAL DESCRIPTION:
;  1235          1294  1 !
;  1236          1295  1 !       BLANK_FILL fills a string with blanks.  It is identical to
;  1237          1296  1 !       a CH$FILL with a first argument of %C' '.  A separate called
;  1238          1297  1 !       routine is used so that registers R0 through R5 are free in
;  1239          1298  1 !       the calling routine.
;  1240          1299  1 !
;  1241          1300  1 ! CALLING SEQUENCE:
;  1242          1301  1 !
;  1243          1302  1 !       pointer.rlu.v = BLANK_FILL (len.rlu.v, addr.wbu.r)
;  1244          1303  1 !
;  1245          1304  1 ! FORMAL PARAMETERS:
;  1246          1305  1 !
;  1247          1306  1 !       len             Number of bytes to blank fill.
;  1248          1307  1 !       addr            Address of string to fill.
;  1249          1308  1 !
;  1250          1309  1 ! IMPLICIT INPUTS:
;  1251          1310  1 !
;  1252          1311  1 !       NONE
;  1253          1312  1 !
;  1254          1313  1 ! IMPLICIT OUTPUTS:
;  1255          1314  1 !
;  1256          1315  1 !       NONE
;  1257          1316  1 !
;  1258          1317  1 ! FUNCTION VALUE:
;  1259          1318  1 !
;  1260          1319  1 !       The address of the next byte past the blank-filled string.
;  1261          1320  1 !
;  1262          1321  1 ! SIDE EFFECTS:
;  1263          1322  1 !
;  1264          1323  1 !       NONE
;  1265          1324  1 !
;  1266          1325  1
;  1267          1326  1 !++
;  1268          1327  2     BEGIN
;  1269          1328  2     RETURN CH$FILL (%C' ', .LEN, .ADDR);
;  1270          1329  1     END;


                                                003C 00000 BLANK_FILL:
                                                           .WORD   Save R2,R3,R4,R5                    ;  1287
    04   AC          20          6E          00  2C 00002           MOVC5   #0, (SP), #32, LEN, @ADDR   ;  1328
                                                08  BC     00008
                                    50          53  D0 0000A           MOVL    R3, R0
                                                04  0000D           RET                                 ;  1329

; Routine Size:  14 bytes,    Routine Base:  _FOR$CODE + 0452
```

```
;  1272          1330  1 ROUTINE MOVE_CHAR (                          ! Move characters
;  1273          1331  1         LEN,                                 ! Fill length
;  1274          1332  1         SOURCE,                              ! Source address
;  1275          1333  1         DEST)                                ! Destination address
;  1276          1334  1     : NOVALUE =
;  1277          1335  1
;  1278          1336  1 !++
;  1279          1337  1 ! FUNCTIONAL DESCRIPTION:
;  1280          1338  1 !
;  1281          1339  1 !       MOVE_CHAR moves characters from one string to another.  It is
;  1282          1340  1 !       identical to CH$MOVE except that it does not return a value.
;  1283          1341  1 !       A separate called routine is used so that registers R0 through
;  1284          1342  1 !       R5 are free in the calling routine.
;  1285          1343  1 !
;  1286          1344  1 ! CALLING SEQUENCE:
;  1287          1345  1 !
;  1288          1346  1 !       CALL MOVE_CHAR (len.rwu.v, source.rbu.r, dest.wbu.r)
;  1289          1347  1 !
;  1290          1348  1 ! FORMAL PARAMETERS:
;  1291          1349  1 !
;  1292          1350  1 !       len             Number of bytes to move.
;  1293          1351  1 !       source          Address of string to move from.
;  1294          1352  1 !       dest            Address of string to move to.
;  1295          1353  1 !
;  1296          1354  1 ! IMPLICIT INPUTS:
;  1297          1355  1 !
;  1298          1356  1 !       NONE
;  1299          1357  1 !
;  1300          1358  1 ! IMPLICIT OUTPUTS:
;  1301          1359  1 !
;  1302          1360  1 !       NONE
;  1303          1361  1 !
;  1304          1362  1 ! FUNCTION VALUE:
;  1305          1363  1 !
;  1306          1364  1 !       NONE
;  1307          1365  1 !
;  1308          1366  1 ! SIDE EFFECTS:
;  1309          1367  1 !
;  1310          1368  1 !       NONE
;  1311          1369  1 !
;  1312          1370  1
;  1313          1371  1 !--
;  1314          1372  2     BEGIN
;  1315          1373  2     CH$MOVE (.LEN, .SOURCE, .DEST);
;  1316          1374  1     END;


                                        003C 00000 MOVE_CHAR:
                                                   .WORD   Save R2,R3,R4,R5                ; 1330
                    0C   BC      08   BC      04   AC  28 00002          MOVC3   LEN, @SOURCE, @DEST     ; 1373
                                                 04 00009          RET                     ; 1374

; Routine Size:  10 bytes,    Routine Base:  _FOR$CODE + 0460
```

```
; 1317          1375  1 END                                   ! End of FOR$$UDF_WF Module
; 1318          1376  1
; 1319          1377  0 ELUDOM



;                                   PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;   _FOR$CODE                     1130  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;   _FOR$DATA                       84  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)


;                                 Library Statistics
;
;                                          -------- Symbols --------      Pages       Processing
;       File                               Total    Loaded   Percent      Mapped      Time
;
;   _$255$DUA28:[SYSLIB]STARLET.L32;1       9776        11         0         581        00:01.1
;   _$255$DUA28:[FORRTL.OBJ]FORLIB.L32;1     711       210        29          52        00:00.5
;   _$255$DUA28:[FORRTL.OBJ]RTLLIB.L32;1      36         0         0           8        00:00.1



;                                       COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS$:FORUDFWF/OBJ=OBJ$:FORUDFWF MSRC$:FORUDFWF/UPDATE=(ENH$:FORUDFWF)

; Size:          1041 code + 173 data bytes
; Run Time:          00:28.7
; Elapsed Time:      01:07.5
; Lines/CPU Min:      2880
; Lexemes/CPU-Min: 17175
; Memory Used:  353 pages
; Compilation Complete
```

FORSTOP
LIS

FORUDFR
LIS

FORTIMEDS
LIS

FORUDFWN
LIS

FORUDFW
LIS

FORUDFRN
LIS

FORUDFRU
LIS

FORTIME
LIS

FORUDFWF
LIS

FORUDFRF
LIS